




Accelerating Markov Chain Model Checking: Good-for-Games Meets Unambiguous Automata

Yong Li^{1,2}, Soumyajit Paul², Sven Schewe², and
Qiyi Tang²



¹ Key Laboratory of System Software (Chinese Academy of Sciences) and SKLCS, Institute of Software, Chinese Academy of Sciences, Beijing, PR. China
liyong@ios.ac.cn

² University of Liverpool, Liverpool, UK
{yong.li3, soumyajit.paul, sven.schewe, qiyi.tang}@liverpool.ac.uk



Abstract. Good-for-Games (GfG) automata require that their nondeterminism can be resolved on-the-fly, while unambiguous automata guarantee that no word has more than one accepting run. These two mutually exclusive ways of restricted nondeterminism play their roles independently in Markov chain model checking (MCMC) for almost a decade but synthesising them seems hopeless: an automaton that is both GfG and unambiguous is essentially deterministic. This work breaks this perception by combining the strengths of unambiguity with the GfG co-Büchi minimisation recently proposed by Abu Radi and Kupferman. More precisely, this combination allows us to turn unambiguous automata to certain types of probabilistic automata that can be used for MCMC. The resulting automata can be exponentially smaller, and we have provided a family of automata exemplifying this state space reduction, which translates into a significant acceleration of MCMC.

Keywords: Unambiguous Büchi automata, Good-for-games automata, Markov chains, Probabilistic model checking

1 Introduction

Markov chains are widely used across numerous application domains, including computer science, engineering, operations research, and the modelling of population growth and behaviour. Verifying Markov chains against ω -regular properties, such as Linear Temporal Logic (LTL) specifications [10] and Linear Dynamic Logic (LDL) [8], has long been the one exemption to the rule that automata are a perfect tool for the analysis of Markovian models. The automata suitable for Markov chain model checking (MCMC) have to restrict their nondeterminism in some way [10]. We consider two known types of restricted nondeterminism for MCMC in this paper, namely good-for-games (GfG) and unambiguous variants. These two types of automata have been applied in MCMC independently for almost a decade.

GfG automata [18] have the ability to resolve nondeterministic choices on-the-fly. That is, a strategy exists for the automaton to generate an accepting run transition by transition, even when an accepting word is given letter by letter, such that the resolution of the nondeterminism does not depend on the future. Klein et al. [21] first proved that GfG automata can be used in probabilistic model checking, and thus in MCMC. Despite this, GfG automata have not been widely adopted in state of the art tools for verifying ω -regular properties, such as ePMC [14], PRISM [23], and IscasMC [16], due to the lack of effective constructions of GfG automata.

Unambiguous automata have been more successful in MCMC than GfG automata. Unambiguous automata restrict the nondeterminism by guaranteeing that every word has at most one accepting run. Unambiguity has gained traction in MCMC when Couvreur et al. [11] developed a polynomial-time algorithm to model check against separated Büchi automata - a class of Büchi automata that have disjoint languages from every pair of states. This reduced the complexity of automata-based LTL model checking down to PSPACE, matching its lower bound [10]. Recently, Baier et al. [5] have developed an NC approach to model check against a more general and succinct class of automata than separated Büchi automata called *unambiguous Büchi automata* (UBAs) [9]. Different to GfG automata, there is also a rich set of sources of UBAs from LTL [25,20], and LDL [24].

Furthermore, unambiguity retains the expressive power of Büchi automata: UBAs can recognise all ω -regular languages [9], while GfG Büchi (or co-Büchi) automata only accept the languages recognised by their deterministic counterparts [18,22]. On one hand, unambiguity retains the expressive power of the ω -regular languages, but poses challenges in minimisation reduction of the automata. On the other hand, GfGness loses expressive power, but transition-based GfG co-Büchi can benefit from a recent polynomial-time minimisation construction [1]. Both types of restricted nondeterminism have been independently studied for MCMC. A natural question arises: *can we combine the strengths of unambiguity and GfGness for MCMC?* A trivial attempt would be to consider unambiguous GfG automata, which actually can be made deterministic by removing unreachable and unproductive states (cf. [6] for details). That is, we would not only lose expressive power in this attempt, but would also have to determinise the automaton.

Contributions. This work is the *first* successful effort to synergise unambiguity and GfGness. We first show that, when the input UBA is deterministic, we can directly apply the GfG co-Büchi minimisation construction to the deterministic automaton. We then prove that the resultant minimised automaton is *stochastically resolvable*, meaning that we can turn it into a probabilistic automaton by resolving the nondeterminism randomly such that the probabilistic automaton accepts (resp. rejects) every word from the input automaton’s language (resp. its complement language) with probability one. This specific probabilistic automaton is well-suited for MCMC. The case for nondeterministic UBA, however, is more challenging because the GfG minimisation algorithm does not work di-

rectly on general UBAs. We address this challenge by proposing an algorithm that accommodates both GfGness and unambiguity for MCMC (Section 4). Furthermore, we provide a family of UBAs for which our proposed minimisation reduction can reduce the state space exponentially. Empirical evaluations on selected benchmarks show that our algorithm significantly accelerates MCMC, thus making our contribution a valuable addition to the existing portfolio of MCMC techniques.

Related Work. Other known types of automata suitable for MCMC are limit-deterministic Büchi automata with mild constraints [15,27] and a more general class called good-for-MDPs (GfM) automata [17,26]. We note that GfMness is a generalisation of GfGness and minimising GfM Büchi automata is PSPACE-hard [26]. Therefore, we do not consider GfM automata in this work. Klein et al. [21] used the algorithm in [18] to construct GfG automata from LTL, which are even larger than their deterministic counterparts. This highlights the need for more efficient methods for constructing GfG automata.

The efficient minimisation algorithm of [1] only works for GfG co-Büchi automata. Our work bypasses this limit and allows for using GfG co-Büchi minimisation in MCMC against all ω -regular properties. Baier et al. [5] also work on MCMC against UBAs. Thanks to our reduction algorithm for UBAs, our model checking algorithm can be (possibly exponentially) more efficient than theirs for certain properties.

Organisation of the Paper. We discuss the preliminaries in Section 2. In Section 3, we present an alternative approach for model checking Markov chains against DBA specifications using GfG automata. Section 4 introduces our algorithm for model checking Markov chains against UBA specifications via GfG automata. Next, in Section 5, we present a case study on a family of UBAs, where our reduction achieves exponential state-space savings, followed by implementation details and experimental results demonstrating the better scalability of our algorithm compared to existing UBA model checking methods. Finally, we conclude in Section 6.

2 Preliminaries

Automata. A nondeterministic automaton is a tuple $\mathcal{A} = (\Sigma, Q, \delta, q_0, \alpha)$ where Σ is the alphabet, Q is a finite set of states, $q_0 \in Q$ is the initial state, $\delta : Q \times \Sigma \mapsto 2^Q$ is the transition function and $\alpha \subseteq Q \times \Sigma \times Q$ is a set of transitions for describing a transition-based acceptance condition. The transition function δ induces a transition relation $\Delta_{\mathcal{A}} \subseteq Q \times \Sigma \times Q$, where for every two states $p, q \in Q$ and letter $a \in \Sigma$, we have that $(p, a, q) \in \Delta_{\mathcal{A}}$ iff $q \in \delta(p, a)$. When it is clear from the context, we omit the subscript \mathcal{A} and directly write Δ . We also extend δ to sets and words in a usual way, by letting $\delta(S, a) = \bigcup_{q \in S} \delta(q, a)$, $\delta(S, \epsilon) = S$ and $\delta(S, u \cdot a) = \delta(\delta(S, u), a)$, where $u \in \Sigma^*$, $a \in \Sigma$ and ϵ is the empty word. Let $\delta^\alpha(p, a) = \{q \mid (p, a, q) \in \alpha\}$ and $\delta^{\bar{\alpha}}(p, a) = \{q \mid (p, a, q) \notin \alpha\}$.

We use the terms α -transitions and $\bar{\alpha}$ -transitions to refer to transitions in α and in $\Delta \setminus \alpha$, respectively.

A word $w \in \Sigma^\omega$ is an infinite sequence of letters in Σ . A run of a word $w = a_0 a_1 \dots \in \Sigma^\omega$ in \mathcal{A} is an infinite sequence $\rho = p_0 p_1 \dots \in Q^\omega$ of states such that $p_0 = q_0$ and $\forall i \in \mathbb{N}, p_{i+1} \in \delta(p_i, a_i)$. A run ρ is accepted by \mathcal{A} if it satisfies the acceptance condition α . We consider both Büchi and co-Büchi acceptance conditions to determine which runs should be accepted. We say that a run ρ satisfies the Büchi acceptance condition if, for infinitely many $i \in \mathbb{N}$, $(p_i, a_i, p_{i+1}) \in \alpha$, while ρ satisfies the co-Büchi acceptance condition if we only have $(p_i, a_i, p_{i+1}) \in \alpha$ for finitely many i . The language of \mathcal{A} , denoted by $\mathcal{L}(\mathcal{A})$, is the set of all words w with at least one accepting run. For $q \in Q$, let \mathcal{A}^q be the automaton obtained from \mathcal{A} by setting q as the initial state.

We write $\bar{\mathcal{L}}$ for the complement language of \mathcal{L} , that is, $\bar{\mathcal{L}} = \Sigma^\omega \setminus \mathcal{L}$. We say that two states $p, q \in Q$ are equivalent, denoted $p \sim_{\mathcal{A}} q$, if $\mathcal{L}(\mathcal{A}^p) = \mathcal{L}(\mathcal{A}^q)$. Let u be a finite word and L be an ω -language. We define the right language of u as $u^{-1}L = \{w \in L \mid uw \in L\}$.

We say that \mathcal{A} is *deterministic* if, for all $q \in Q, a \in \Sigma$, we have $|\delta(q, a)| \leq 1$. \mathcal{A} is said to be *unambiguous* if there is *at most* one accepting run in \mathcal{A} for every word $w \in \Sigma^\omega$. \mathcal{A} is said to have a *diamond* if there exist two states $q, q' \in Q$ and a finite word $w \in \Sigma^*$, such that w has two different runs from q to q' in \mathcal{A} . Otherwise, \mathcal{A} is called *diamond-free*. An unambiguous automaton can be made diamond-free in polynomial time. We use DBA/NBA/UBA for deterministic/nondeterministic/unambiguous Büchi automata and DCA/NCA for deterministic/nondeterministic co-Büchi automata.

An automaton \mathcal{A} is called *good-for-games* (GfG) [18]³ if there is a strategy function $f : \Sigma^* \mapsto Q$ such that, for every accepting word $w = a_0 a_1 \dots$, the run $\rho_f = f(\epsilon)f(a_0) \dots f(a_0 \dots a_i) \dots$ is an accepting run. We usually represent a strategy function as a resolver automaton $\mathcal{R} = (\mathbb{M}, m_0, g)$, where \mathbb{M} is the set of memory states of the resolver, $m_0 \in \mathbb{M}$ is the initial memory state and $g : \mathbb{M} \times Q \times \Sigma \mapsto \mathbb{M} \times Q$ is the *deterministic* transition function that selects the successor for a state $q \in Q$ and $a \in \Sigma$. As usual, we also extend g to words such that $g(m_0, q_0, ua) = g(g(m_0, q_0, u), a)$ for all $u \in \Sigma^*$ and $a \in \Sigma$, where $(m_0, q_0) = g(m_0, q_0, \epsilon)$. Consequently, for all finite words $u_1, u_2 \in \Sigma^*$, if $g(m_0, q_0, u_1) = g(m_0, q_0, u_2)$, then $f(u_1) = f(u_2)$.

In [1], a polynomial-time algorithm is presented that, given a GfG-NCA \mathcal{A} , computes a minimal GfG-NCA \mathcal{A}' such that $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$. The minimal GfG-NCA \mathcal{A}' satisfies several important properties. For instance, \mathcal{A}' is semantically deterministic, meaning that, for a state $q \in Q$ in \mathcal{A}' and a letter $a \in \Sigma$, all the a -successors of q (i.e., all states in $\delta(q, a)$) are equivalent. \mathcal{A}' is also *safe deterministic*, meaning all transitions are deterministic *or* α -transitions. That is, for every state $q \in Q$ and letter $a \in \Sigma$, we have that $|\delta(q, a)| \leq 1$ or $\delta^\alpha(q, a) = \delta(q, a)$.

Components. Let $G = (V, E)$ be a directed graph. For $C \subseteq V$ and $E' \subseteq E$ let (C, E') denote a subgraph of G where every edge in E' lies within C . A

³ For ω -automata, GfG automata are equivalent to history-deterministic automata.

component (C, E') is a strongly connected subgraph of G , i.e., for every two vertices $u, v \in C$, there is a path from u to v via E' . A strongly connected component (SCC) (C, E') of G is a maximal, strongly connected subgraph of G , i.e., there does not exist a strongly connected subgraph of G , (C', E'') , such that $C \subsetneq C'$ or $E' \subsetneq E''$. An SCC (C, E) is called a *bottom* SCC (BSCC) if there is no path in G from any $u \in C$ to another SCC (C', E') . The automaton \mathcal{A} induces a directed graph $G_{\mathcal{A}} = (V, E)$ with $V = Q$, and $(p, q) \in E$ iff for some $a \in \Sigma$, $q \in \delta(p, a)$. The SCCs (or components) of \mathcal{A} are in fact those of $G_{\mathcal{A}}$. A *safe* component of \mathcal{A} is an SCC of the graph $G_{\mathcal{A}}^{\bar{a}} = (Q, E^{\bar{a}})$, where $(p, q) \in E^{\bar{a}}$ iff there exists a letter $a \in \Sigma$ such that $q \in \delta^{\bar{a}}(p, a)$.

Markov Chains and Probability Measures. For a finite set S , we denote column vectors by boldface letters such as $\mathbf{x} \in \mathbb{R}^S$, and write \mathbf{x}^\top for the transpose (a row vector) of \mathbf{x} ; in particular, $\mathbf{1} \in \{1\}^S$ and $\mathbf{0} \in \{0\}^S$ are column vectors whose entries are all 1 and 0, respectively. We denote the vector entry for $s \in S$ as $\mathbf{x}(s)$ and write \mathbf{x}_C for the restriction of \mathbf{x} to $C \subseteq S$. Let $\text{Distr}(S)$ be the set of all probability distributions on S . We denote the Dirac distribution concentrated at some $s \in S$ by ι_s .

We consider Markov chains where transitions are labelled with letters from the alphabet Σ of the automaton. A *labelled Markov chain* (LMC) is a tuple $\mathcal{M} = \langle S, \Sigma, M, s_0 \rangle$ where S is a finite and non-empty set of states, the alphabet is Σ (labels), $M : \Sigma \rightarrow [0, 1]^{S \times S}$ is the transition function such that $\sum_a M(a)$ is stochastic⁴ and $s_0 \in S$ is the initial state. \mathcal{M} can be perceived as a labelled weighted graph (WG) with S as the set of states and for $a \in \Sigma$, $M(a) \in [0, 1]^{S \times S}$ is the weighted adjacency matrix corresponding to transitions with label a . A labelled path $\pi = s_0 a_0 s_1 a_1 s_2 \dots$ in \mathcal{M} generates the word $w(\pi) = a_0 a_1 \dots$. We can talk about probability measures of measurable subsets of Σ^ω in the σ -algebra generated by basic cylinder sets. For a finite word $x = a_0, \dots, a_n$, the cylinder set $\text{Cyl}(x) = x\Sigma^\omega$ is the set of all infinite words with x as prefix. The probability measure of the set $\text{Cyl}(x)$ is given by $\Pr(\text{Cyl}(x)) = \iota_{s_0}^\top M(a_0)M(a_1) \dots M(a_n)\mathbf{1}$. Essentially, $\Pr(\text{Cyl}(x))$ is the probability that x is generated by a path of length n in \mathcal{M} . This can be extended to all measurable sets generated by the cylinder sets in the standard way. In particular ω -regular languages are measurable sets. Given \mathcal{M} and an ω -regular language \mathcal{L} , let $\Pr_{\mathcal{M}}(\mathcal{L})$ be the probability that a word generated randomly by \mathcal{M} is in \mathcal{L} . We omit \mathcal{M} when it is clear from context. For an automaton state $q \in Q$ and a state $s \in S$ in \mathcal{M} , $\Pr_s(\mathcal{L}(\mathcal{A}^q))$ is the probability that a word generated by \mathcal{M} is in the language $\mathcal{L}(\mathcal{A}^q)$. We require the LMC be *separated*. This means that for any two distinct states s, s' , and for any word $w \in \Sigma^*$, if there exists a run from s on w , then there does not exist a run from s' on w , and vice versa. This is a mild constraint. As demonstrated in [5], we can satisfy this requirement by making the name of each state part of the alphabet. Alternatively, since we have letters on transitions, we can make the name of the state part of the letter on each outgoing transition.

⁴ A nonnegative matrix is stochastic (resp. substochastic) if each row is stochastic (resp. substochastic), that is, each row adds up to exactly one (resp. at most one).

We also consider general weighted graphs \mathcal{W} over the state space S and matrix W . For $C, D \subseteq S$ we write $W^{C,D}$ for the submatrix of W obtained by deleting the rows not indexed by C and the columns not indexed by D . Similarly, we write $W^{(C,E)}$, where (C, E) is an SCC in the graph of the submatrix of $W^{C,C}$.

In this paper we are interested in the MCMC problem: given an LMC \mathcal{M} and a UBA \mathcal{U} with initial states s_0 and q_0 , respectively, we want to compute the probability $Pr_{s_0}(\mathcal{L}(\mathcal{U}^{q_0}))$. In the following, we begin with the DBA specifications in Section 3, and then proceed to the more involved UBA specifications in Section 4.

3 Model Checking against DBAs

Let $\mathcal{M} = (S, \Sigma, M, s_0)$ be an LMC and $\mathcal{D} = (\Sigma, Q, \delta, q_0, \alpha)$ be a complete deterministic automaton. The classic MCMC algorithm first builds the product LMC $\mathcal{D} \times \mathcal{M}$ and then computes the probability of reaching the set K , denoted $Pr_{\mathcal{D} \times \mathcal{M}}(\Diamond K)$, where K is the set of all states in the *accepting BSCCs* of $\mathcal{D} \times \mathcal{M}$. Accepting BSCCs of $\mathcal{D} \times \mathcal{M}$ are BSCCs with at least one edge whose projection onto \mathcal{D} lies in α . The probability $Pr_{\mathcal{M}}(\mathcal{L}(\mathcal{D}))$ is equal to the reachability probability of accepting BSCCs in $\mathcal{D} \times \mathcal{M}$.

In this section, we propose an alternative procedure to compute this probability by leveraging GfG automata minimisation. Our main idea is to construct an intermediate *probabilistic automaton* (PA) that preserves the language of \mathcal{D} and analyse its product with \mathcal{M} . A PA $\mathcal{P} = (\Sigma, Q, \delta, q_0, \alpha)$ is a nondeterministic automaton equipped with a randomised transition function $\delta : Q \times \Sigma \mapsto \text{Distr}(Q)$, where from state q with letter a , transition to q' is taken with probability $\delta(q, a)(q')$. We often abuse the notation by writing $\delta(q, a)$ to denote its support. Each word $w \in \Sigma^\omega$ induces a probability measure $Pr_{\mathcal{P}}^w$ on Q^ω in the usual way. The probability that \mathcal{P} accepts w , denoted by $Pr_{\mathcal{P}}(w)$, is the probability measure of all accepting runs of w on \mathcal{P} , that is, $Pr_{\mathcal{P}}(w) = Pr_{\mathcal{P}}^w(\{\pi \mid \pi \text{ is an accepting run of } w\})$. For a detailed introduction to PAs, please refer to [3].

Moreover, we require our PAs to be *0/1-PA*. A PA \mathcal{P} is *0/1-PA* if, for any word $w \in \Sigma^\omega$, we have either $Pr_{\mathcal{P}}(w) = 1$ or $Pr_{\mathcal{P}}(w) = 0$. For a *0/1-PA* \mathcal{P} , and with a slight abuse of notation, we say that a word w is accepted by \mathcal{P} (i.e., $w \in \mathcal{L}(\mathcal{P})$) if $Pr_{\mathcal{P}}(w) = 1$. In this paper, we focus on *0/1-PAs* that accept ω -regular languages.

In general, a *0/1-PA* \mathcal{P} may accept a non- ω -regular language [3, Example 4.2.1]. Note that for some $w \notin \mathcal{L}(\mathcal{P})$, there may exist accepting runs in \mathcal{P} , but the probability measure of such accepting runs is 0. Additionally, we assume that \mathcal{P} is complete, meaning that for all $w \in \Sigma^\omega$, we have $Pr_{\mathcal{P}}^w(\{\pi \mid \pi \text{ is a run over } w\}) = 1$.

Given an LMC \mathcal{M} , and state s in \mathcal{M} , let $Pr_s(\mathcal{L}(\mathcal{P}^q))$ be the probability that a random word generated in \mathcal{M} starting from s lies in $\mathcal{L}(\mathcal{P}^q)$. We will first describe how we can compute the probabilities $Pr_s(\mathcal{L}(\mathcal{P}^q))$ by solving a system of linear equations on the product of \mathcal{P} and \mathcal{M} . Given a *0/1-PA* $\mathcal{P} =$

$(\Sigma, Q, \delta, q_0, \alpha)$, we define the product $\mathcal{P} \times \mathcal{M} = (\Sigma, Q \times S, \delta_\otimes, \langle q_0 s_0 \rangle, \alpha_\otimes)$ where $Q \times S$ is the set of states, $\langle q_0 s_0 \rangle \in Q \times S$ is the initial state, $\alpha_\otimes \subseteq \Delta_{\mathcal{P} \times \mathcal{M}}$ is the set of α -transitions of $\mathcal{P} \times \mathcal{M}$ such that $(\langle qs \rangle, a, \langle q' s' \rangle) \in \alpha_\otimes$ if $(q, a, q') \in \alpha$, and $\delta_\otimes : (Q \times S) \times \Sigma \mapsto \text{Distr}(Q \times S)$ is the transition function such that $\delta_\otimes(\langle qs \rangle, a)(\langle q' s' \rangle) = M(a)(s, s') \cdot \delta(q, a)(q')$ for all $\langle qs \rangle, \langle q' s' \rangle \in Q \times S$ and $a \in \Sigma$.

The definition of the product can be easily adapted when \mathcal{M} is a weighted graph. We show in Proposition 2 that \mathcal{P} can be used to model check \mathcal{M} by analysing the product $\mathcal{P} \times \mathcal{M}$. The product $\mathcal{P} \times \mathcal{M}$ can be effectively represented by the matrix $B_\otimes \in \mathbb{R}^{(Q \times S) \times (Q \times S)}$, where $B_\otimes(\langle qs \rangle, \langle q' s' \rangle) = \sum_{a \in \Sigma} \delta_\otimes(\langle qs \rangle, a)(\langle q' s' \rangle)$ for all $\langle qs \rangle, \langle q' s' \rangle \in Q \times S$.

It is not hard to see that the matrix B_\otimes is stochastic and $\mathcal{P} \times \mathcal{M}$ is a Markov chain. The linear equation system in Eq. (1) has a unique solution, denoted by χ . A proof of uniqueness can be found in [4, Theorem 10.19]. The value $\chi(\langle qs \rangle)$ precisely represents the probability of $\langle qs \rangle$ reaching an accepting BSCC.

$$\begin{aligned} \mathbf{x} &= B_\otimes \mathbf{x} \\ \text{for all states } c \text{ in accepting BSCCs} & \quad \mathbf{x}(c) = 1 \\ \text{for all states } c \text{ in rejecting BSCCs} & \quad \mathbf{x}(c) = 0 \end{aligned} \tag{1}$$

Now we establish some useful properties of 0/1-PAs: they are semantically deterministic and can be complemented by simply negating the acceptance condition, similar to deterministic automata.

Lemma 1. *Let \mathcal{P} be a 0/1-PA.*

- (1) *For any $a \in \Sigma$, $q \in Q$ and $q_1, q_2 \in \delta(q, a)$, we have $\mathcal{L}(\mathcal{P}^{q_1}) = \mathcal{L}(\mathcal{P}^{q_2})$.*
- (2) *Let \mathcal{P}' be the PA obtained by negating the acceptance condition. Then \mathcal{P}' is also a 0/1-PA. Moreover, $\mathcal{L}(\mathcal{P}') = \mathcal{L}(\mathcal{P})$.*

Using Lemma 1(1), we show:

Proposition 2. *Let \mathcal{M} be an LMC, and \mathcal{P} be a 0/1-PA recognising the DBA language L . Then $\text{Pr}_s(\mathcal{L}(\mathcal{P}^q)) = \chi(\langle qs \rangle)$ for all $\langle qs \rangle \in Q \times S$. In particular, $\text{Pr}_{\mathcal{M}}(L) = \text{Pr}_{\mathcal{M} \times \mathcal{P}}(\Diamond K)$ where K is the set of all states in the accepting BSCCs of $\mathcal{P} \times \mathcal{M}$.*

Proposition 2 is a simple yet effective generalisation of existing similar standard result for deterministic automata [4] since DBAs and DCAs can be seen as 0/1-PAs by transitioning to the only successor with probability one.

Our source of 0/1-PA is a property satisfied by the minimised GfG-NCA automata in [1] called *stochastic resolvability*. Stochastically resolvable automata can be turned into language-preserving 0/1-PAs by resolving nondeterminism using good (positional) *stochastic resolvers*. A good stochastic resolver $R : Q \times \Sigma \mapsto \text{Distr}(Q)$ for nondeterministic automaton \mathcal{A} is simply a randomised transition function that turns \mathcal{A} into a 0/1-PA, denoted by $\mathcal{A} \times R$, such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A} \times R)$. A stochastic resolver is said to be uniform when for every q and a , $R(q, a)$ is the uniform distribution over $\delta(q, a)$.

With the preparations above, we now propose the following steps to model check an LMC \mathcal{M} against a DBA \mathcal{D} : (1) regard (complete) DBA \mathcal{D} as a DCA \mathcal{C} , (2) minimise \mathcal{C} to a GfG-NCA \mathcal{G} using [1], (3) resolve the nondeterminism by random choices and obtain a 0/1 PA \mathcal{P} with Büchi acceptance condition s.t. $\mathcal{L}(\mathcal{D}) = \mathcal{L}(\mathcal{P})$, (4) compute the product $\mathcal{P} \times \mathcal{M}$, and (5) calculate the reachability probability of accepting BSCCs by solving Eq. (1). The 0/1-PA \mathcal{P} is expected to be smaller than \mathcal{D} to make our approach more efficient compared to the classic MCMC against DBAs.

The main observation is that the minimal GfG-NCA \mathcal{G} produced by [1] has a very simple good stochastic resolver R , the uniform stochastic resolver. This makes \mathcal{G} a stochastically resolvable automaton. Then, by Lemma 1(2), we can obtain a 0/1-PA of $\mathcal{L}(\mathcal{D})$ by interpreting the α -transitions as accepting in \mathcal{G} .

Lemma 3. *Let \mathcal{G} be an NCA that is semantically deterministic and safe deterministic and let R be a uniform stochastic resolver for \mathcal{G} . Then, the PA $\mathcal{G} \times R$ with Büchi acceptance condition is a 0/1-PA. Also, $\mathcal{L}(\mathcal{G} \times R) = \mathcal{L}(\mathcal{D})$.*

We sketch our proof idea as follows. Let $w \in \Sigma^\omega$. First, assume that $w \in \mathcal{L}(\mathcal{D})$, i.e., $w \notin \mathcal{L}(\mathcal{G})$. It immediately follows that $Pr_{\mathcal{G} \times R}(w) = 1$ since all runs of \mathcal{G} over w are rejecting. Now we assume that $w \notin \mathcal{L}(\mathcal{D})$, i.e. $w \in \mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{C})$. Since \mathcal{G} is safe deterministic, every accepting run of \mathcal{G} over a word $w \in \mathcal{L}(\mathcal{C})$ will eventually get trapped in a deterministic safe component. Therefore, the probability measure of every accepting run of \mathcal{G} over w is positive because as soon as the run enters the deterministic safe component, all visited transitions have probability one. Assume by contradiction that $Pr_{\mathcal{G} \times R}(w) = Pr_{\mathcal{G} \times R}^w(\{\pi \mid \pi \text{ is rejecting run over } w \text{ in } \mathcal{G}\}) > 0$. This means that there exists a state $q \in \delta(q_0, u)$ such that the probability measure of rejecting runs of w' in \mathcal{G} from q is 1 where $w = uw'$. However, since \mathcal{G} is semantically deterministic and $w \in \mathcal{L}(\mathcal{G})$, we have $w' \in \mathcal{L}(\mathcal{G}^q)$. This indicates that there is a run of \mathcal{G} over w' from q that goes to a deterministic safe component and stays there. But then this entails that the probability measure of accepting runs over w' in \mathcal{G} from q is positive, leading to contradiction. Hence, $Pr_{\mathcal{G} \times R}(w) = 0$. We can conclude that \mathcal{G} is stochastically resolvable and $\mathcal{L}(\mathcal{D}) = \mathcal{L}(\mathcal{G} \times R)$.

Let $\mathcal{P} = \mathcal{G} \times R$ where R is the uniform stochastic resolver for \mathcal{G} . With Lemma 3 and Proposition 2, our main result of this section immediately follows.

Theorem 4. *Given an LMC \mathcal{M} and a DBA \mathcal{D} , we have $Pr_{\mathcal{M}}(\mathcal{L}(\mathcal{D})) = \chi(q_0)$ where q_0 is the initial state of $\mathcal{P} \times \mathcal{M}$ and χ is the unique solution of the linear equation system Eq. (1).*

According to [21], GfG automata can be used to model check LMCs by constructing the product of \mathcal{M} and \mathcal{G} , where the nondeterminism in \mathcal{G} is resolved by incorporating additional actions. This process yields a product Markov decision process (MDP) rather than a Markov chain. The probability $Pr_{\mathcal{M}}(\mathcal{L}(\mathcal{D}))$ can then be expressed as one minus the maximum satisfiability of $\mathcal{L}(\mathcal{G})$ within the resulting MDP. Nonetheless, our algorithm still provides some advantage in this

context, as it reduces the computational complexity from solving a linear programming problem for MDPs to solving a linear system of equations for LMCs. Beyond this advantage, the MCMC algorithm against DBA specifications paves the way towards our primary objective—MCMC against UBA specifications.

4 Model Checking against UBAs

The DBA languages cannot express all ω -regular properties. This section considers UBAs, which can represent all ω -regular properties, as specifications. Our MCMC algorithm against UBAs leverages similar idea for the DBA case which also constructs a PA \mathcal{P} using the minimisation algorithm in [1]. The first challenge we face here is that the minimisation construction only works on GfG-NCAs but our input automaton \mathcal{U} is unambiguous. If we assume that \mathcal{U} is both GfG and unambiguous, then the automaton is essentially deterministic⁵ [6, Proposition 9]; this again will lose expressiveness.

We address the first challenge by determinising the UBA \mathcal{U} with auxiliary letters. That is, we make the nondeterministic choices explicit by marking each choice with different fresh letters. This way, we obtain a complete DBA \mathcal{D} over an extended alphabet Σ' , and regard it as a DCA \mathcal{C} . Since \mathcal{C} is deterministic, we can again compute a minimal GfG-NCA \mathcal{G} accepting $\mathcal{L}(\mathcal{C})$. Clearly, \mathcal{G} has a positional stochastic resolver according to Lemma 3. However, \mathcal{G} is defined over Σ' , which is different than the alphabet Σ in the LMC \mathcal{M} . Our second challenge is how to define the product of \mathcal{M} and \mathcal{P} . To resolve this, we turn \mathcal{M} into a weighted graph \mathcal{W} over Σ' in a way that the product $\mathcal{G} \times \mathcal{W}$ can guide \mathcal{G} to obtain a positional stochastic resolver, yielding the product $\mathcal{P} \times \mathcal{W}$. When we map Σ' back to Σ in the product $\mathcal{P} \times \mathcal{W}$, we can still calculate the correct probability $Pr_{\mathcal{M}}(\mathcal{L}(\mathcal{U}))$.

Let $\mathcal{M} = (S, \Sigma, M, s_0)$ be the given LMC and $\mathcal{U} = (\Sigma, Q, \delta, q_0, \alpha)$ the given UBA. We dedicate the rest of this section to describe in detail how we address the two challenges above and present our new MCMC algorithm.

4.1 Product Construction

We first describe our determinisation approach for \mathcal{U} . This approach preserves the underlying graph of \mathcal{U} but modifies the alphabet along with providing means to recover the original language. In theory, we can obtain a DBA $\mathcal{D} = \langle \Sigma', Q, q_0, \delta_{\mathcal{D}}, \alpha_{\mathcal{D}} \rangle$ over new alphabet $\Sigma' \subseteq \Sigma \times \mathbb{N}$ from \mathcal{U} , using any map $\mathbf{p} : \Delta_{\mathcal{U}} \mapsto \Delta_{\mathcal{D}}$ such that any two transitions $(q_1, a, q'_1), (q_1, a, q'_2)$ in $\Delta_{\mathcal{U}}$ with $q'_1 \neq q'_2$, are mapped by \mathbf{p} to transitions in $\Delta_{\mathcal{D}}$ with $\langle a, i \rangle$ and $\langle a, j \rangle$, that is, two distinct labels from Σ' . This gives a natural projection map $\mathbf{p}^{-1} : \Sigma' \mapsto \Sigma$, that is, $\mathbf{p}^{-1}(\langle a, i \rangle) = a$ for all $\langle a, i \rangle \in \Sigma'$, which can be naturally extended to infinite words and languages.

⁵ GfG unambiguous automata are determinisable by pruning, meaning they can be seen as deterministic automata with additional transitions on top.

For a given state $q \in Q$, a letter $a \in \Sigma$, without loss of generality we can assume an ordering on the successors states $\delta(q, a)$, i.e. $\delta(q, a) = \{q'_0, \dots, q'_h\}$. We define $\mathbf{p}((q, a, q'_i)) = (q, a', q'_i)$ where $a' = \langle a, i \rangle \in \Sigma'$. With $\alpha_{\mathcal{D}} = \mathbf{p}(\alpha_{\mathcal{U}})$, it is easy to see that $\mathcal{L}(\mathcal{U}) = \mathbf{p}^{-1}(\mathcal{L}(\mathcal{D}))$. Essentially in the transitions of \mathcal{D} we have encoded the information regarding which successor was taken in \mathcal{U} . Hence a word in $\mathcal{L}(\mathcal{D})$ uniquely determines its origin word in $\mathcal{L}(\mathcal{U})$ obtained by applying \mathbf{p}^{-1} . This gives us the following crucial observation on how \mathcal{D} retains information on the unambiguity of \mathcal{U} .

Lemma 5. *For two distinct words $w_1, w_2 \in \mathcal{L}(\mathcal{D})$, it always holds that $\mathbf{p}^{-1}(w_1) \neq \mathbf{p}^{-1}(w_2)$.*

We assume \mathcal{D} is complete and can also be read as a DCA \mathcal{C} with $\mathcal{L}(\mathcal{C}) = \overline{\mathcal{L}(\mathcal{D})}$.

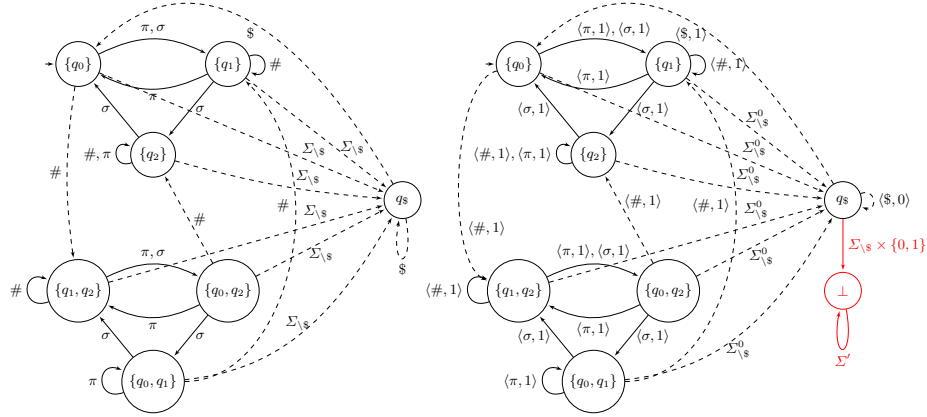


Fig. 1: **Left:** An example UBA \mathcal{U}_3 with alphabet $\Sigma = \{\pi, \sigma, \#, \$\}$; let $\Sigma_{\setminus \$} = \{\pi, \sigma, \#\}$. The dashed transitions are α -transitions which are accepting for Büchi automata. **Right:** Determinise the UBA one the left into a complete DBA with $\Sigma' = \Sigma \times \{0, 1\}$. Let $\Sigma_{\setminus \$}^0 = \Sigma_{\setminus \$} \times \{0\}$.

Example 6. The UBA \mathcal{U}_3 (shown on the left of Fig. 1) belongs to a family of UBAs $\{\mathcal{U}_i\}_i$ for which our algorithm produces an exponentially smaller product. Since there are at most two nondeterministic choices at every state in \mathcal{U}_3 , we can resolve the nondeterminism with a boolean variable. A value of 0 (false) will direct transitions to the state q_s , whereas a value of 1 will take the transition that doesn't lead to q_s . The result is a DBA \mathcal{D}_3 over $\Sigma' = \Sigma \times \{0, 1\}$; see the right of Fig. 1. Since we will later treat \mathcal{D}_3 as a DCA accepting the complement language, we make it complete by adding a new sink state \perp . Specifically, we add non-accepting $\langle \$, 0 \rangle$ - and $\langle \$, 1 \rangle$ -transitions from q to \perp for $q \in 2^{\{q_0, q_1, q_2\}}$ and for the missing letters from q_s to \perp , and a non-accepting self-loop on \perp for all letters in Σ' . Transitions from $q \in 2^{\{q_0, q_1, q_2\}}$ to \perp are omitted in the figure.

The projection \mathbf{p}^{-1} , which maps every letter of Σ' to a letter of Σ , is defined as follows: $\mathbf{p}^{-1}(\langle a, i \rangle) = a$ for all $a \in \Sigma$ and $i \in \{0, 1\}$. \square

By applying the minimisation algorithm from [1] on \mathcal{C} , we obtain a minimal GfG-NCA $\mathcal{G} = \langle \Sigma', Q_{\mathcal{G}}, q_{0_{\mathcal{G}}}, \delta_{\mathcal{G}}, \alpha_{\mathcal{G}} \rangle$ that recognises the same language as \mathcal{C} . Let R be a uniform stochastic resolver for \mathcal{G} . By Lemma 3, the PA \mathcal{P} with a Büchi acceptance condition, defined as $\mathcal{P} = \mathcal{G} \times R$, is a 0/1-PA, and its language satisfies $\mathcal{L}(\mathcal{P}) = \mathcal{L}(\mathcal{D})$. Formally, \mathcal{P} differs from \mathcal{G} only in its transition function, which is given by $\delta_{\mathcal{P}}(q, a)(q') = \frac{1}{|\delta_{\mathcal{G}}(q, a)|}$ for all $(q, a, q') \in \Delta_{\mathcal{G}}$. We have:

Lemma 7. $\mathcal{L}(\mathcal{U}) = \mathbf{p}^{-1}(\mathcal{L}(\mathcal{D})) = \mathbf{p}^{-1}(\mathcal{L}(\mathcal{P}))$.

Being a 0/1-PA, \mathcal{P} also enjoys the properties given in Lemma 1. That is, \mathcal{P} is semantically deterministic and can be complemented by simply complementing the acceptance condition. To show that \mathcal{P} can be used to model check LMCs, we first establish additional properties of \mathcal{P} concerning its projected languages over Σ . Firstly, since \mathcal{P} is semantically deterministic, for $\langle a, i \rangle \in \Sigma'$ and a state q in \mathcal{P} , projected languages of two $\langle a, i \rangle$ -successors of q would also be the same. Secondly, let $\langle a, i \rangle, \langle a, j \rangle \in \Sigma'$ be two distinct letters. Let $q \in Q_{\mathcal{G}}$ be a reachable state, and $q_1 \in \delta_{\mathcal{P}}(q, \langle a, i \rangle)$ and $q_2 \in \delta_{\mathcal{P}}(q, \langle a, j \rangle)$ be two successors. Since q is reachable, for some $w \in \Sigma'^*$ having a run from initial state to q , we have $w\langle a, i \rangle \mathcal{L}(\mathcal{P}^{q_1}) \cup w\langle a, j \rangle \mathcal{L}(\mathcal{P}^{q_2}) \subseteq \mathcal{L}(\mathcal{P}) = \mathcal{L}(\mathcal{D})$. From Lemma 5 it follows that $\mathbf{p}^{-1}(\mathcal{L}(\mathcal{P}^{q_1})) \cap \mathbf{p}^{-1}(\mathcal{L}(\mathcal{P}^{q_2})) = \emptyset$. Hence we have the following lemma.

Lemma 8. *Let $q \in Q_{\mathcal{G}}$, $\langle a, i \rangle$ and $\langle a, j \rangle$ be two different letters in Σ' . We have:*

- (1) *for all states $q_1, q_2 \in \delta_{\mathcal{P}}(q, \langle a, i \rangle)$, it holds that $\mathbf{p}^{-1}(\mathcal{L}(\mathcal{P}^{q_1})) = \mathbf{p}^{-1}(\mathcal{L}(\mathcal{P}^{q_2}))$*
- (2) *for states $q_1 \in \delta_{\mathcal{P}}(q, \langle a, i \rangle)$ and $q_2 \in \delta_{\mathcal{P}}(q, \langle a, j \rangle)$, it holds that $\mathbf{p}^{-1}(\mathcal{L}(\mathcal{P}^{q_1})) \cap \mathbf{p}^{-1}(\mathcal{L}(\mathcal{P}^{q_2})) = \emptyset$.*

Example 9. Consider the complete DBA \mathcal{D}_3 in Example 6. We now read it as a DCA \mathcal{C}_3 . Since DCAs are GfG, we can apply the algorithm from [1] to obtain a minimal GfG-NCA \mathcal{G}_3 ; see left of Fig. 2. We then turn the NCA into a 0/1-PA \mathcal{P} by turning all transitions to randomised ones. Note that ignoring the probabilities and projecting Σ' down to Σ does *not* result in an automaton that is either unambiguous or diamond-free. E.g. $\#\#\$$ creates three paths from q_0 back to q_0 , and $(\#\#\$)^\omega$ has an uncountable set of accepting runs. \square

Lemma 8 is a key result of this paper, enabling us to combine good-for-gameness and unambiguity without losing the expressive power of ω -regular languages. The main idea is to leverage the good-for-game properties to resolve nondeterminism in the outer layer of the language over Σ' , while exploiting the unambiguity inherent in the inner layer of the language over Σ . This is facilitated by the mapping function \mathbf{p} .

To align the LMC \mathcal{M} with the alphabet Σ' of \mathcal{P} , we transform \mathcal{M} , originally defined over Σ , into a weighted graph (WG) \mathcal{W} with alphabet Σ' . Let $\mathcal{W} = \langle S, \Sigma', W, s_0 \rangle$, where $W(a) = M(\mathbf{p}^{-1}(a))$ for all $a \in \Sigma'$.

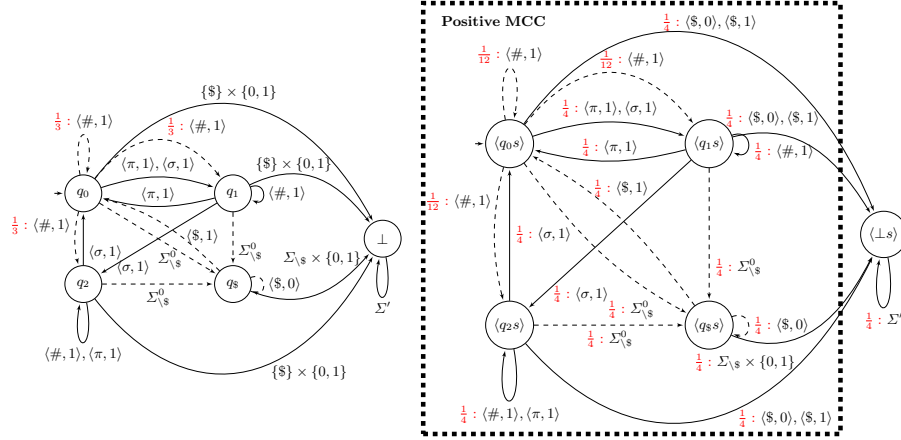


Fig. 2: **Left:** Read the complete DBA in Example 6 as a DCA and minimise it into a GfG-NCA \mathcal{G}_3 using the algorithm from [1]. The fractions in orange should be disregarded when reading it as an NCA. We then turn it into a 0/1-PA \mathcal{P} by applying a stochastic resolver on \mathcal{G}_3 . **Right:** The product of the GfG-NCA \mathcal{G}_3 and the WG \mathcal{W} (or the PA \mathcal{P} and the WG \mathcal{W}). There is one positive MCC in this product which is highlighted in thick dashed box.

Example 10. The LMC $\mathcal{M} = \langle S, \Sigma, M, s \rangle$ generates infinite words over $\Sigma = \{\sigma, \pi, \#, \$\}$ uniformly at random, with a single state s and $M(a)(s, s) = \frac{1}{4}$ for all $a \in \Sigma$ (see left of the figure below). Since there is only one state in the LMC, it is obviously separated. Transforming \mathcal{M} into a WG $\mathcal{W} = \langle S, \Sigma', W, s \rangle$ over $\Sigma' = \Sigma \times \{0, 1\}$ preserves the state space, and $W(a')(s, s) = \frac{1}{4}$ for all $a' \in \Sigma'$ (see right of the figure below). \square



With both challenges being addressed, we are ready to define the product WG $\mathcal{P} \times \mathcal{W} = (\Sigma', Q_\otimes, \delta_{\mathcal{P} \times \mathcal{W}}, \langle q_0 \mathcal{G} s_0 \rangle, \alpha_\otimes)$, the state space Q_\otimes of which is equal to $Q_\mathcal{G} \times S$. As usual, $(\langle qs \rangle, a, \langle q's' \rangle) \in \alpha_\otimes$ if $(q, a, q') \in \alpha_\mathcal{G}$.

This product can actually be seen as the product of \mathcal{P} and \mathcal{M} where there is a transition from $\langle qs \rangle$ to $\langle q's' \rangle$ over a letter $a \in \Sigma'$ if $M(\mathbf{p}^{-1}(a))(s, s') > 0$ and $(q, a, q') \in \Delta_\mathcal{P}$. It is important to use the maps \mathbf{p} and \mathbf{p}^{-1} in the model checking procedure to associate the language $\mathcal{L}(\mathcal{D})$ with the language $\mathcal{L}(\mathcal{U})$, in particular, deriving the corresponding linear equations.

We define the matrix $B_\otimes \in \mathbb{R}^{Q_\otimes \times Q_\otimes}$, where $B_\otimes(\langle qs \rangle, \langle q's' \rangle) = \sum_{a \in \Sigma'} \delta_{\mathcal{P} \times \mathcal{W}}(\langle qs \rangle, a)(\langle q's' \rangle)$ for all $\langle qs \rangle, \langle q's' \rangle \in Q_\otimes$. Unlike the matrix in

Eq. (1), the matrix B_\otimes here may not be stochastic. We conclude this subsection by constructing the basic linear equation system $\mathbf{x} = B_\otimes \mathbf{x}$ to compute the probability $Pr_{\mathcal{M}}(\mathcal{L}(\mathcal{U}))$. Hereafter, we use $\chi(\langle qs \rangle)$ to denote the probability $Pr_s(\mathbf{p}^{-1}(\mathcal{L}(\mathcal{P}^q)))$ for simplicity. Based on the relationship between the languages of automata (Lemma 7), we have $\chi(\langle q_0 s_0 \rangle) = Pr_{s_0}(\mathbf{p}^{-1}(\mathcal{L}(\mathcal{P}))) = Pr_{s_0}(\mathcal{L}(\mathcal{U})) = Pr_{\mathcal{M}}(\mathcal{L}(\mathcal{U}))$. Using Lemma 8 and that the LMC is separated, we have the following.

Proposition 11. *Let \mathcal{M} be an LMC and \mathcal{U} be a UBA. The vector χ satisfies $\mathbf{x} = B_\otimes \mathbf{x}$.*

Proposition 11 only provides a basic linear equation system, but it does not answer the following two questions: how to identify accepting components, and how to make sure that the obtained linear equation system has a unique solution. We address these questions in the remainder of this section.

4.2 Analysing MCCs of the Product

In the DBA case, we add additional equations—one for each BSCC—to ensure a unique solution (cf. Eq. (1)). Since the underlying matrix is stochastic, it is straightforward that, for any state in a BSCC, if an outgoing transition labelled with a belongs to the BSCC, then all other outgoing transitions labelled with a must also be within the BSCC; otherwise, the SCC is transient. In the UBA case, however, since the matrix B_\otimes may *not* be stochastic, rather than looking at BSCCs, we focus on the components of $\mathcal{P} \times \mathcal{W}$ that are *closed* and *maximal*, which we refer to as maximal closed components (MCCs). We call a component (C, E) closed if, for any state $q \in C$ and any letter $a \in \Sigma'$, either *all* or *none* of a -successors of q lie in C , i.e. $\delta(q, a) \subseteq C$ or $\delta(q, a) \cap C = \emptyset$. A closed component (C, E) is maximal if there does not exist a closed component (C', E') such that $C \subsetneq C'$ or $E \subsetneq E'$. Alg. 1 computes MCCs efficiently.⁶

Algorithm 1: MCC Computation

```

1 repeat
2   compute SCCs of  $\mathcal{P} \times \mathcal{W}$ ;
3   if there exist an SCC  $(C, E)$ , a state  $\langle qs \rangle \in C$  and a letter  $a \in \Sigma'$  such
      that  $(\langle qs \rangle, a, \langle q's' \rangle) \in \Delta_{\mathcal{P} \times \mathcal{W}}$  and  $\langle qs \rangle, \langle q's' \rangle$  belong to different SCCs
4     then
5       remove all transitions labelled with  $a$  from  $\langle qs \rangle$  in  $C$ ;
6 until no further changes.
```

An MCC (C, E) is called *positive* if it is accepting, meaning that it contains some accepting transition, and *recurrent* if there exists a positive solution \mathbf{x}

⁶ MCC is closely related to maximal end components (MECs) in Markov decision processes (MDPs). Alg. 1 is an adaption to MCC of the standard algorithm for computing MECs in MDPs, see e.g. [4, Algorithm 47].

to the following linear equation system: $B_{\otimes}^{(C,C)} \mathbf{x} = \mathbf{x}$ and $\mathbf{x}(c) = \mathbf{x}(d)$ for all $c, d \in C$ such that c and d can be reached simultaneously from some state by the same word. States c and d are language equivalent when $\mathcal{P} \times \mathcal{W}$ is interpreted as a Büchi automaton, which entails from Lemma 8. In case the underlying matrix B_{\otimes} is stochastic, the positive MCCs are accepting BSCCs. Thus, in the DBA case, all accepting BSCCs are positive MCCs.

For a state $\langle qs \rangle$ in $\mathcal{P} \times \mathcal{W}$ that cannot reach a positive MCC, the probability $\chi(\langle qs \rangle)$ is zero. Consequently, we can add the equations $\mathbf{x}(\langle qs \rangle) = 0$ to our linear equation system for all such states $\langle qs \rangle$ that cannot reach a positive MCC.

Proposition 12. *For a state $\langle qs \rangle$ of $\mathcal{P} \times \mathcal{W}$ that cannot reach a positive MCC, we have $\chi(\langle qs \rangle) = 0$.*

We can also show that the probability that the UBA \mathcal{U} accepts a word randomly generated by the LMC \mathcal{M} is positive if, and only if, a positive MCC exists in $\mathcal{P} \times \mathcal{W}$. Hence, for qualitative model checking, it suffices to check whether there is a positive MCC.

Proposition 13. *(Qualitative MCMC) $\Pr_{\mathcal{M}}(\mathcal{L}(\mathcal{U})) > 0$ iff the product $\mathcal{P} \times \mathcal{W}$ has a positive MCC.*

For positive MCCs (C, E) of $\mathcal{P} \times \mathcal{W}$, the key observation is that each positive MCC includes all transitions within it and can, therefore, be referred to simply as C . This also entails that the projection of a positive MCC onto the states of the LMC is a BSCC of the LMC. Moreover, there exist sets called cuts that can be used to compute the values $\chi(\langle qs \rangle)$ for the states $\langle qs \rangle$. Cuts were introduced in [5] for MCMC against UBA specifications. Intuitively, these cuts are subsets $K \subseteq C$ of states with pairwise disjoint languages over Σ , such that almost all words have an accepting run starting from some state $q \in K$. We denote the transition function of $\mathcal{P} \times \mathcal{W}$ over Σ by $\delta_{\otimes} : Q_{\otimes} \times \Sigma \rightarrow 2^{Q_{\otimes}}$, defined as follows: $q' \in \delta_{\otimes}(q, a)$ if and only if there exists $a' \in \Sigma'$ such that $\mathbf{p}^{-1}(a') = a$ and $\delta_{\mathcal{P} \times \mathcal{W}}(q, a')(q') > 0$. For an MCC C , we write δ_{\otimes}^C to denote the restriction of δ_{\otimes} to C .

A subset $K \subseteq C$ is a *cut* for a positive MCC C if it satisfies the following conditions: (1) $K \subseteq \delta_{\otimes}(c, v)$ for some $c = \langle qs \rangle \in C$ and $v \in \Sigma^*$, (2) $\delta_{\otimes}(K, w) \neq \emptyset$ for all $w \in \Sigma^*$ such that w can be generated in \mathcal{M} from some s where $\langle qs \rangle \in K$, and (3) all states in K are language disjoint.⁷

Given a cut $K \subseteq C$, we call its characteristic vector $\mu_C \in \{0, 1\}^C$ a cut vector, that is, $\mu_C(k) = 1$ iff $k \in K$. A cut can be computed in polynomial time in the size of the positive MCC C using Alg. 2. Our cut computation is inspired by [5]; however, our algorithm additionally accounts for language equivalence of states, $\sim_{\mathcal{P} \times \mathcal{W}}$ where $\mathcal{P} \times \mathcal{W}$ is read as a Büchi automaton, mainly because the product $\mathcal{P} \times \mathcal{W}$ is neither unambiguous nor diamond-free (cf. Example 9).

The following lemma summarises the key properties of cuts that we will need.

Lemma 14. *Let C be a positive MCC. Then*

⁷ The cuts defined in [5] did not require (3), as any set K satisfying (1) would automatically satisfy (3) in their product due to its unambiguity and diamond-freeness.

Algorithm 2: Computing a cut K of a positive MCC (C, E)

```

1  $c \in C$  ( $c$  can be any state in  $C$ );
2  $w := \varepsilon$  (the empty word);
3 while  $\exists v \in \Sigma^*$  and  $d \not\sim_{\mathcal{P} \times \mathcal{W}} c$  with  $\{c, d\} \subseteq \delta_{\otimes}^C(c, v)$  and  $\delta_{\otimes}^C(d, w) \neq \emptyset$  do
4    $w := vw$ ;
5 return  $K = \delta_{\otimes}^C(c, w) / \sim_{\mathcal{P} \times \mathcal{W}}$ .
```

- (1) C has a cut $K \subseteq C$ and $\mu_C^\top \chi_C = 1$, that is, $\sum_{k \in K} \chi(k) = 1$.
(2) One can compute a cut K for C in polynomial time.

Now, we are ready to present the complete linear equation system:

$$\begin{aligned}
& \mathbf{x} = B_{\otimes} \mathbf{x} \\
& \text{for all positive MCCs } C \text{ of } \mathcal{P} \times \mathcal{W} : \quad \mu_C^\top \mathbf{x}_C = 1 \\
& \text{[where } \mu_C \text{ is a cut vector for the positive MCC } C\text{]} \\
& \text{for all states } c \text{ that cannot reach a positive MCC:} \quad \mathbf{x}(c) = 0 \quad (2)
\end{aligned}$$

By combining Propositions 11, 12, and 13, and Lemma 14, we can demonstrate that Eq. (2) has a unique solution \mathbf{v} , which corresponds to χ . Consequently, we can prove Alg. 3, which summarises our approach to model checking LMCs against UBA specifications, correctly computes the satisfying probability.

Theorem 15. (*Quantitative MCMC*) *Given an LMC \mathcal{M} and a UBA \mathcal{U} , the linear equation system Eq. (2) has a unique solution \mathbf{v} . Moreover, we have $\mathbf{v} = \chi$, and thus $\text{Pr}_{\mathcal{M}}(\mathcal{L}(\mathcal{U})) = \mathbf{v}(q_0)$, where q_0 denotes the initial state in $\mathcal{P} \times \mathcal{W}$.*

Algorithm 3: Summary of the overall algorithm

```

1 Determine the UBA  $\mathcal{U}$  into a complete DBA  $\mathcal{D}$  on the new alphabet  $\Sigma'$ ;
   Read  $\mathcal{D}$  as a DCA  $\mathcal{C}$ ;
2 Obtain the minimal GfG-NCA  $\mathcal{G}$  which is language equivalent to  $\mathcal{C}$  by running
   the algorithm in [1];
3 Turn the LMC  $\mathcal{M}$  into a WG  $\mathcal{W}$  on  $\Sigma'$ ;
4 Obtain a 0/1-PA  $\mathcal{P}$  by applying a positional stochastic resolver on  $\mathcal{G}$ ;
5 Build the product  $\mathcal{P} \times \mathcal{W}$ , and denote the underlying matrix as  $B_{\otimes}$ ;
6 return  $\chi(q_0)$  where  $q_0$  is the initial state of  $\mathcal{P} \times \mathcal{W}$  and  $\chi$  is the unique
   solution of the linear equation system Eq. (2).
```

Example 16. The product of the PA \mathcal{P} in Example 9 and the WG \mathcal{W} in Example 10 is shown on the right of Fig. 2. There is a single accepting MCC (C, E) highlighted in this product, which consists of all four states of the product

$\langle q_0s \rangle, \langle q_1s \rangle, \langle q_2s \rangle, \langle q_{\$}s \rangle$ and all transitions between them. We have $B_{\otimes}^{(C,C)}\mathbf{x} = \mathbf{x}$ and $\mathbf{x}(\langle q_0s \rangle) = \mathbf{x}(\langle q_1s \rangle) = \mathbf{x}(\langle q_2s \rangle)$, which indicates that this MCC is recurrent. Thus, the probability is positive that the given UBA \mathcal{U}_3 accepts a word randomly generated by the LMC \mathcal{M} . This is indeed the case, as the probability is $\frac{3}{4}$. This is because the probability that a randomly generated word in \mathcal{M} has finitely many $\$$ is 0. And among the words containing infinitely many $\$$, the set of words rejected by the UBA \mathcal{U}_3 is exactly $\{\$w \mid w \in \Sigma^\omega\}$ - all ω -words that start with $\$$. These words are generated by the LMC \mathcal{M} with a probability of $\frac{1}{4}$. The probability is determined by adding an equation for the *cut* of the positive MCC, $\mu_C^\top \mathbf{x}_C = 1$, and an equation $\mathbf{x}(\langle \perp s \rangle) = 0$ for states that cannot reach a positive MCC. A cut $K = \{\langle q_i s \rangle, \langle q_{\$} s \rangle\}$ for any $i \in \{0, 1, 2\}$, thus a cut vector, can be computed by running Alg. 2. \square

Correctness Proofs. For proof of correctness, we establish the existence of a good and fair resolver \mathcal{R} for the product $\mathcal{G} \times \mathcal{W}$. This enables the construction of the product WG $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ over Σ' and $\mathcal{B}_{\otimes'}$ over Σ .

The product $\mathcal{G} \times \mathcal{W}$ differs from $\mathcal{P} \times \mathcal{W}$ only in terms of its transition function. The resolver is termed *good* for $\mathcal{G} \times \mathcal{W}$ because, for every word $w \in \mathcal{L}(\mathcal{G})$ generated by \mathcal{W} , the run of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ on w visits an α -transition only finitely often. It is *fair* as, for every ω -word $w = a_0 a_1 \dots$, if a transition $(\langle qsm \rangle, a, \langle q's'm' \rangle) \in \Delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$ has been visited infinitely often, then, for all transitions $(\langle qs \rangle, a, \langle q''s' \rangle) \in \Delta_{\mathcal{G} \times \mathcal{W}}$, some transition $(\langle qsm'' \rangle, a, \langle q''s'm''' \rangle) \in \Delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$ must be visited infinitely often as well. In our approach, the stochastic resolver takes the place of the good and fair resolver. It is *good* since it “preserves” the language of $\mathcal{U} \times \mathcal{M}$ when projecting $\mathcal{P} \times \mathcal{W}$ back to the original alphabet, and it is *fair* in the sense that if an a -transition of a state is taken infinitely often, then all its a -transitions must be taken infinitely often with probability one.

The final product WG, $\mathcal{B}_{\otimes'}$ over Σ , is both diamond-free and unambiguous, enabling us to adapt the algorithm from [5] for both qualitative and quantitative model checking. This allows us to use the structural similarity between $\mathcal{B}_{\otimes'}$ and $\mathcal{P} \times \mathcal{W}$ to establish the correctness of our approach. The full proof is presented in the full version.

5 Experiments

In this section, we present experimental results from our implementation of our algorithm for model checking LMCs against UBAs and compare our implementation against the one in [5]. Section 5.1 describes results validating exponential speed-up of our algorithm using a family of UBAs, for which the resulting automata are exponentially smaller after GfG minimisation. Section 5.2 compares the sizes of automata before and after automata transformation with benchmarks from [5], highlighting how our proposed algorithm effectively reduces the state sizes of automata across these benchmarks.

5.1 Case Study: A Family of UBAs with Exponential Gain

We substantiate our claims of exponential speed up in Markov chain model checking by providing a family of UBAs for which our algorithm produces an exponentially smaller product graph in the final stage of the algorithm. The UBA \mathcal{U}_3 in this family is illustrated on the left of Fig. 1, while the GfG-NCA \mathcal{G}_3 is on the left of Fig. 2.

Theorem 17. *There exists a family of UBAs $\{\mathcal{U}_n\}_{n \geq 2}$ with $|\mathcal{U}_n| = \Omega(2^n)$ such that the GfG automata \mathcal{G}_n produced as a sub-routine of Alg. 3 has size $O(n)$.*

Below, we define this family of UBAs and then give a brief description of the languages they accept. The construction of these UBAs are inspired from the family of DCAs in [22]. A UBA $\mathcal{U}_n = (\Sigma, Q_n, \delta_n, \{q_0\}, \alpha_n)$ over the alphabet $\Sigma = \{\sigma, \pi, \#, \$\}$ with initial state $\{q_0\}$ in the family consists of

- set of states $Q_n = \{q_\$ \} \cup 2^{B_n}$ for $B_n = \{q_0, \dots, q_{n-1}\}$;
- the transition function δ_n producing the set of transitions $\Delta_n \subseteq Q_n \times \Sigma \times Q_n$:
 $\sigma: (P, \sigma, P') \in \Delta_n$ for $P' = \{q_{(i+1) \bmod n} \mid q_i \in P\}$,
 $\pi: (P, \pi, P') \in \Delta_n$ for $P' = \{q_i \mid q_i \in P \wedge i \geq 2\} \cup \{q_{(i+1) \bmod 2} \mid q_i \in P \wedge i < 2\}$,
 $\#: (P, \#, P') \in \Delta_n$ for P' is $\{q_1, \dots, q_{n-1}\}$ if $P = \{q_0\}$ and $P \setminus \{q_0\}$ otherwise,
 $\$: (P, a, q_\$) \in \Delta_n \forall P \in 2^{B_n}, a \in \Sigma \setminus \$$ and $(q_\$, \$, \{q_0\}), (q_\$, \$, q_\$) \in \Delta_n$;
- the set of transitions $\alpha_n \subseteq \Delta_n$ describing the acceptance condition: $\alpha_n = \{(P_1, \#, P_2) \in \Delta_n \mid q_0 \in P_1\} \cup \{(P, a, q_\$) \mid P \in 2^{B_n}\} \cup \{(q_\$, \$, \{q_0\}), (q_\$, \$, q_\$)\}$.

Similar to [22], each letter in Σ represents an action performed using an infinite set of tokens and a set B_n of n boxes. Initially, each box has exactly one token. σ moves tokens cyclically, i.e. moves token from q_i to $q_{(i+1) \bmod n}$ for every i ; π swaps the tokens in boxes q_0 and q_1 ; $\#$ discards the token in box q_0 , replacing with a new one; and $\$$ discards tokens from all boxes and replaces with new ones. The language $\mathcal{L}(\mathcal{U}_n)$ is the set of all infinite sequences of actions such that any token present in some box at any instance is eventually discarded. The automaton \mathcal{U}_n operates by tracking tokens within possible sets of boxes using states $P \subseteq 2^{B_n}$ or by predicting a $\$$ action through the designated state $q_\$$. More details and proofs can be found in the full version.

Implementation and Experiments. We have implemented a probabilistic model-checking procedure for LMCs and UBA specifications using the algorithm described in Section 4. This implementation extends the probabilistic model checker PRISM [23]. This enables a comparison with the implementation of the algorithm presented in [5, Section 8], which we refer to as PRISM UBA. All experiments were carried out on a computer with one Apple M3 8-core CPU with 16GB of RAM running MacOS and a time limit of 30 minutes. Our implementation consists of two components: the first is a stand-alone tool that performs UBA determinisation and GfG-NCA minimisation (lines 1–2 of Alg. 3), and the second is our model-checking algorithm (starting from line 3 of Alg. 3), which takes the GfG-NCA generated by the first component and the input LMC. The first component is implemented in C++ and built on top of the Spot automata library [13]: it takes as input a UBA in HOA format [2],

Table 1: Comparison of PRISM UBA from [5] (rank-based) and our algorithm for model checking against a family of UBAs ($n \in \{3, \dots, 8\}$) on a randomly generated LMC with 20 states. The table shows the number of states in the automata, product sizes, and model checking times (t_{total}). For our algorithm, t_{total} includes t_{tr} (UBA determinisation and GfG minimisation, lines 1–2 of Alg. 3) and t_{mc} (model checking the minimal GfG automaton, starting at line 3). A dash (‘-’) indicates a timeout (30 min) or a stack overflow (1 GB).

n	PRISM UBA [5]			Our Algorithm				
	UBA	Product	t_{total}	GfG	Product	t_{total}	t_{tr}	t_{mc}
3	7	100	0.324 s	5	100	0.384 s	0.052 s	0.332 s
4	15	200	0.882 s	6	120	0.434 s	0.021 s	0.413 s
5	31	400	7.220 s	7	140	0.471 s	0.077 s	0.394 s
6	63	800	87.527 s	8	160	0.513 s	0.059 s	0.454 s
7	127	1600	1201.652 s	9	180	0.583 s	0.117 s	0.466 s
8	255	3200	-	10	200	0.513 s	0.102 s	0.411 s

determinises the UBA to a DCA using the approach from Section 4.1, minimises the DCA using [1], and outputs the minimal GfG-NCA in HOA format.

Similar to PRISM UBA, the second component is also based on the explicit engine of PRISM, where the Markov chain is represented explicitly. Our implementation supports direct verification against a path specification given by a UBA provided in the HOA format [2]. For the linear algebra parts of the algorithms, we use the COLT library [19], as also used in [5].

We consider direct model checking against UBA specifications, where the UBAs are taken from the family described earlier in this section. The experiments were conducted on a simple randomly generated LMC. Table 1 presents the results for $n \in \{3, \dots, 8\}$, obtained with a timeout of 30 minutes and a stack size of 1 GB. These results demonstrate that, for this family of UBAs, our algorithm is competitive with the UBA model checking algorithm from [5], particularly for $n \geq 4$, as highlighted in the table. Notably, PRISM UBA becomes unsuccessful for $n \geq 8$, unable to model check against the UBAs due to the immense size of the products. In contrast, our algorithm successfully scales to larger instances within a reasonable time for model checking. We are aware of a more efficient iterative version of the PRISM-UBA algorithm proposed in [5, Section 8.1]. We believe that a similar iterative optimisation could be applied to our algorithm as well, which we leave for future work.

5.2 Additional Benchmarks

Here we consider benchmarks from [5] and report whether the GfG minimisation step reduces the size of automata, since reduction in the number of states in the input UBA accelerates the subsequent model-checking procedure.

Complete and Nearly-complete UBAs. First, we consider the two families of parametrised UBAs, the complete and the nearly-complete UBAs bench-

marks, from [5, Section 8.2]. Our minimisation algorithm usually achieves a significant reduction in the number of states, often cutting the state count by more than half. The following table compares the sizes of UBAs and the GfG automata, where k is a parameter for the two families of UBAs.

Complete UBAs				Nearly-complete UBAs			
k	UBA	GfG	t_{tr}	k	UBA	GfG	t_{tr}
5	193	96	0.121 s	5	193	94	0.122 s
6	449	192	0.126 s	6	449	190	0.126 s
7	1025	384	0.215 s	7	1025	382	0.251 s
8	2305	768	1.119 s	8	2305	766	1.070 s
9	5121	1536	8.160 s	9	5121	1534	8.179 s

LTL Specifications for Bounded Retransmission Protocol. Next, we consider the two LTL properties described in [5, Section 8.3] for the bounded retransmission protocol (BRP) model in the PRISM benchmark suite [23]. BRP concerns a single message transmission and retrying for a bounded number of times in case of an error. The first formula is $\varphi^k = (\neg ack_received) \mathcal{U} retransmit \wedge (\neg ack_received \mathcal{U}^{=k} ack_received)$, where $a \mathcal{U}^{=k} b$ stands for $a \wedge \neg b \wedge \bigcirc(a \wedge \neg b) \wedge \dots \wedge \bigcirc^{k-1}(a \wedge \neg b) \wedge \bigcirc^k b$. φ^k ensures that the message was retransmitted k steps before acknowledgement. The second formula is $\psi^k = (msg_send \rightarrow (ack_send \wedge \Diamond^{\leq k} ack_received))$, where $\Diamond^{\leq k}$ denotes $a \vee \bigcirc(a \vee \bigcirc(\dots \vee \bigcirc a))$ (repeated k times). ψ^k ensures that for every message sent, the receiver of the message sends an acknowledgement, and this acknowledgement is received within the next k steps.

For both benchmarks, using Spot 2.12.1 (the version used for our experiments), we observed that after some value of k , the generated UBAs were very large, whereas the GfG automata obtained from our minimisation step were significantly smaller. For example, for $k = 14$ in the first LTL family, the UBA produced by Spot 2.12.1 had 6,147 states on our macOS machine whereas our tool obtained GfG with only 16 states. For $k = 12$ in the second LTL family, the UBA had 8,452 states, while our tool obtained a GfG automaton with 159 states within a few seconds. Thus, the advantage of our model-checking algorithm over [5] becomes clear when the UBAs are significantly reduced after some value of k . In fact, the first BRP LTL formulas define a classic family of languages where DBAs can be exponentially larger than the corresponding UBAs where minimal UBAs for k should have $k + 2$ states. Our minimisation approach for $k = 14$ gives GfG automaton with 16 (i.e. $k + 2$) states, matching the size of the minimal UBA. We note that [5] used Spot 2.7, and also reported that the UBA had size 16 for $k = 14$. This then demonstrates that our algorithm is not only effective but also more robust across different tool choices.

6 Conclusion

We have synergised two recent advancements in the efficient verification of Markov chains: facilitating unambiguous Büchi automata and minimising GfG-

NCAAs [1]. These two classes of automata are very surprising candidates for obtaining synergistic effects, because they generalise deterministic automata in very different ways. Unambiguous automata guarantee the uniqueness of an accepting run, which has been at the heart of the approach to use them in model checking Markov chains [5]. Broadly speaking, it allows for *not* resolving the nondeterminism and to still measure the accepting paths. This heavily relies on two things: one is that this is the only nondeterminism to consider (which is why it does not work for MDPs), and the other is that the path to acceptance is narrow, allowing for only a single accepting run of the automaton on every word of the Markov chain in the target language.

It is fair to say that GfG nondeterminism is the polar opposite. It needs to be resolvable *on-the-fly* and we need to choose between transitions to various language equivalent states. Locally—i.e., on any finite prefix of a run—all of these transitions can be taken; broadly speaking, one could say that it depends on the path to acceptance being broad: having infinitely many accepting runs for a word is the norm. To justify the term polar opposite further, these two restrictions of nondeterminism are mutually exclusive ways to generalise deterministic automata in that unambiguous GfG automata are essentially deterministic (cf. [6]).

The second big difference between these classes is that UBAs recognise all ω -regular languages, while GfG-NCA can only recognise co-Büchi languages. We have shown that these concepts can nevertheless be brought together in an intricate construction, where the nondeterminism of the automaton is resolved randomly. When interpreting the probabilistic transitions as nondeterministic, the resulting PA is not guaranteed to be unambiguous. Yet, the resulting probabilities can be used in MCMC and retain the full expressive power of UBAs.

This allows us to reduce the state space of a UBA in polynomial time, where the target is not normally a UBA, and the optimisation step minimises a DCA as a GfG-NCA, an exponentially more succinct automata class [22]. We believe that our algorithm can further speed up model checking against LTL [12,5], LDL and weak alternating automata [24], and UBAs in general, especially when the input LMCs are large, as is often the case with real-world models.

Acknowledgments. We would like to thank the anonymous reviewers for their suggestions that helped improve the paper. This work has been supported in part by the Engineering and Physical Sciences Research Council (EPSRC) through grant EP/X03688X/1 and EP/X042596/1, ISCAS Basic Research (Grant Nos. ISCAS-JCZD-202406, ISCAS-JCZD-202302), CAS Project for Young Scientists in Basic Research (Grant No. YSBR-040), and ISCAS New Cultivation Project ISCAS-PYFX-202201.

Disclosure of Interests. The authors have no competing interests.

References

1. Abu Radi, B., Kupferman, O.: Minimization and canonization of GFG transition-based automata. *Log. Methods Comput. Sci.* **18**(3), 16:1–16:33 (2022). [https://doi.org/10.46298/LMCS-18\(3:16\)2022](https://doi.org/10.46298/LMCS-18(3:16)2022)
2. Babiak, T., Blahoudek, F., Duret-Lutz, A., Klein, J., Křetínský, J., Müller, D., Parker, D., Strejček, J.: The Hanoi omega-automata format. In: Kroening, D., Păsăreanu, C.S. (eds.) *Computer Aided Verification*. pp. 479–486. Springer International Publishing, Cham (2015)
3. Baier, C., Größer, M., Bertrand, N.: Probabilistic ω -automata. *J. ACM* **59**(1), 1:1–1:52 (2012). <https://doi.org/10.1145/2108242.2108243>
4. Baier, C., Katoen, J.: *Principles of model checking*. MIT Press (2008)
5. Baier, C., Kiefer, S., Klein, J., Müller, D., Worrell, J.: Markov chains and unambiguous automata. *J. Comput. Syst. Sci.* **136**, 113–134 (2023). <https://doi.org/10.1016/J.JCSS.2023.03.005>
6. Boker, U., Kupferman, O., Skrzypczak, M.: How deterministic are good-for-games automata? In: Lokam, S.V., Ramanujam, R. (eds.) *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, December 11–15, 2017, Kanpur, India*. LIPIcs, vol. 93, pp. 18:1–18:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017). <https://doi.org/10.4230/LIPICS.FSTTCS.2017.18>
7. Büchi, J.R.: State-strategies for games in F G. *J. Symb. Log.* **48**(4), 1171–1198 (1983). <https://doi.org/10.2307/2273681>
8. Bustan, D., Rubin, S., Vardi, M.Y.: Verifying omega-regular properties of Markov chains. In: Alur, R., Peled, D.A. (eds.) *Computer Aided Verification, 16th International Conference, CAV 2004, Boston, MA, USA, July 13–17, 2004, Proceedings*. *Lecture Notes in Computer Science*, vol. 3114, pp. 189–201. Springer (2004). https://doi.org/10.1007/978-3-540-27813-9_15
9. Carton, O., Michel, M.: Unambiguous Büchi automata. *Theor. Comput. Sci.* **297**(1–3), 37–81 (2003). [https://doi.org/10.1016/S0304-3975\(02\)00618-7](https://doi.org/10.1016/S0304-3975(02)00618-7)
10. Courcoubetis, C., Yannakakis, M.: The complexity of probabilistic verification. *J. ACM* **42**(4), 857–907 (1995). <https://doi.org/10.1145/210332.210339>
11. Couvreur, J., Saheb, N., Sutre, G.: An optimal automata approach to LTL model checking of probabilistic systems. In: Vardi, M.Y., Voronkov, A. (eds.) *Logic for Programming, Artificial Intelligence, and Reasoning, 10th International Conference, LPAR 2003, Almaty, Kazakhstan, September 22–26, 2003, Proceedings*. *Lecture Notes in Computer Science*, vol. 2850, pp. 361–375. Springer (2003). https://doi.org/10.1007/978-3-540-39813-4_26
12. Duret-Lutz, A., Lewkowicz, A., Fauchille, A., Michaud, T., Renault, E., Xu, L.: Spot 2.0 - A framework for LTL and omega-automata manipulation. In: Artho, C., Legay, A., Peled, D. (eds.) *Automated Technology for Verification and Analysis - 14th International Symposium, ATVA 2016, Chiba, Japan, October 17–20, 2016, Proceedings*. *Lecture Notes in Computer Science*, vol. 9938, pp. 122–129 (2016). https://doi.org/10.1007/978-3-319-46520-3_8
13. Duret-Lutz, A., Renault, E., Colange, M., Renkin, F., Aisse, A.G., Schlehuber-Caissier, P., Medioni, T., Martin, A., Dubois, J., Gillard, C., Lauko, H.: From Spot 2.0 to Spot 2.10: What’s new? In: *Proceedings of the 34th International Conference on Computer Aided Verification (CAV’22)*. *Lecture Notes in Computer Science*, vol. 13372, pp. 174–187. Springer (Aug 2022). https://doi.org/10.1007/978-3-031-13188-2_9

14. Fu, C., Hahn, E.M., Li, Y., Schewe, S., Sun, M., Turrini, A., Zhang, L.: EPMC gets knowledge in multi-agent systems. In: Finkbeiner, B., Wies, T. (eds.) Verification, Model Checking, and Abstract Interpretation - 23rd International Conference, VMCAI 2022, Philadelphia, PA, USA, January 16-18, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13182, pp. 93–107. Springer (2022). https://doi.org/10.1007/978-3-030-94583-1_5
15. Hahn, E.M., Li, G., Schewe, S., Turrini, A., Zhang, L.: Lazy probabilistic model checking without determinisation. In: Aceto, L., de Frutos-Escrig, D. (eds.) 26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1-4, 2015. LIPIcs, vol. 42, pp. 354–367. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2015). <https://doi.org/10.4230/LIPICS.CONCUR.2015.354>
16. Hahn, E.M., Li, Y., Schewe, S., Turrini, A., Zhang, L.: ISCASMC: A web-based probabilistic model checker. In: Jones, C.B., Pihlajasaari, P., Sun, J. (eds.) FM 2014: Formal Methods - 19th International Symposium, Singapore, May 12-16, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8442, pp. 312–317. Springer (2014). https://doi.org/10.1007/978-3-319-06410-9_22
17. Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: Good-for-MDPs automata for probabilistic analysis and reinforcement learning. In: Biere, A., Parker, D. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12078, pp. 306–323. Springer (2020). https://doi.org/10.1007/978-3-030-45190-5_17
18. Henzinger, T.A., Piterman, N.: Solving games without determinization. In: Ésik, Z. (ed.) Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4207, pp. 395–410. Springer (2006). https://doi.org/10.1007/11874683_26
19. Hoschek, W.: The Colt distribution: Open source libraries for high performance scientific and technical computing in java. URL: <http://nicewww.cern.ch/hoschek/colt/index.htm> (2002)
20. Jantsch, S., Müller, D., Baier, C., Klein, J.: From LTL to unambiguous Büchi automata via disambiguation of alternating automata. Formal Methods Syst. Des. **58**(1-2), 42–82 (2021). <https://doi.org/10.1007/S10703-021-00379-Z>
21. Klein, J., Müller, D., Baier, C., Klüppelholz, S.: Are good-for-games automata good for probabilistic model checking? In: Dediu, A., Martín-Vide, C., Sierra-Rodríguez, J.L., Truthe, B. (eds.) Language and Automata Theory and Applications - 8th International Conference, LATA 2014, Madrid, Spain, March 10-14, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8370, pp. 453–465. Springer (2014). https://doi.org/10.1007/978-3-319-04921-2_37
22. Kuperberg, D., Skrzypczak, M.: On determinisation of good-for-games automata. In: Halldórsson, M.M., Iwama, K., Kobayashi, N., Speckmann, B. (eds.) Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9135, pp. 299–310. Springer (2015). https://doi.org/10.1007/978-3-662-47666-6_24
23. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) Proc. 23rd Interna-

- tional Conference on Computer Aided Verification (CAV'11). LNCS, vol. 6806, pp. 585–591. Springer (2011)
24. Li, Y., Schewe, S., Vardi, M.Y.: Singly exponential translation of alternating weak Büchi automata to unambiguous Büchi automata. *Theor. Comput. Sci.* **1006**, 114650 (2024). <https://doi.org/10.1016/J.TCS.2024.114650>
 25. Rohde, G.S.: Alternating automata and the temporal logic of ordinals. Ph.D. thesis, University of Illinois at Urbana-Champaign (1997)
 26. Schewe, S., Tang, Q., Zhanabekova, T.: Deciding what is good-for-MDPs. In: Pérez, G.A., Raskin, J. (eds.) 34th International Conference on Concurrency Theory, CONCUR 2023, September 18–23, 2023, Antwerp, Belgium. LIPIcs, vol. 279, pp. 35:1–35:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2023). <https://doi.org/10.4230/LIPICS.CONCUR.2023.35>
 27. Sickert, S., Esparza, J., Jaax, S., Kretínský, J.: Limit-deterministic Büchi automata for linear temporal logic. In: Chaudhuri, S., Farzan, A. (eds.) Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17–23, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9780, pp. 312–332. Springer (2016). https://doi.org/10.1007/978-3-319-41540-6_17

A Missing proofs in Section 3

Lemma 1. *Let \mathcal{P} be a 0/1-PA.*

- (1) *For any $a \in \Sigma$, $q \in Q$ and $q_1, q_2 \in \delta(q, a)$, we have $\mathcal{L}(\mathcal{P}^{q_1}) = \mathcal{L}(\mathcal{P}^{q_2})$.*
- (2) *Let \mathcal{P}' be the PA obtained by negating the acceptance condition. Then \mathcal{P}' is also a 0/1-PA. Moreover, $\mathcal{L}(\mathcal{P}') = \overline{\mathcal{L}(\mathcal{P})}$.*

Proof. (1) *Claim.* For $u \in \Sigma^*$, $w \in \Sigma^\omega$ and $q \in \delta(q_0, u)$, we have $Pr_{\mathcal{P}^q}(w) = 1$ if $uw \in L$ and $Pr_{\mathcal{P}^q}(w) = 0$ if $uw \notin L$.

First, for a word $aw \in \Sigma^\omega$, we have $Pr_{\mathcal{P}^q}(aw) = \sum_{q' \in \delta(q, a)} \delta(q, a)(q') Pr_{\mathcal{P}^{q'}}(w)$. So clearly, if $Pr_{\mathcal{P}^q}(aw) = 0$, then $Pr_{\mathcal{P}^{q'}}(w) = 0$ for all $q' \in \delta(q, a)$. We now prove the case when $Pr_{\mathcal{P}^q}(aw) = 1$. Assume by contradiction that for some successor $q' \in \delta(q, a)$, we have $Pr_{\mathcal{P}^{q'}}(w) < 1$. Then clearly, $Pr_{\mathcal{P}^q}(aw) = \sum_{q' \in \delta(q, a)} \delta(q, a)(q') Pr_{\mathcal{P}^{q'}}(w) < 1$, leading to contradiction. The claim then holds.

The following claim can then be proved by repeatedly applying the first claim from the initial state q_0 . This completes the proof of item (1) of the lemma.

Claim. For $a \in \Sigma$ and $w \in \Sigma^\omega$, if $Pr_{\mathcal{P}^q}(a \cdot w) = 1$, then $Pr_{\mathcal{P}^{q'}}(w) = 1$ and if $Pr_{\mathcal{P}^q}(a \cdot w) = 0$, then $Pr_{\mathcal{P}^{q'}}(w) = 0$ for all $q' \in \delta(q, a)$.

- (2) Let ϕ be the acceptance condition. Let $w \in \Sigma^\omega$ and Π^w be the set of runs in \mathcal{P} over w . Let Π_{acc}^w be the set of accepting runs and Π_{rej}^w be the set of rejecting runs. That is, for every run $\rho \in \Pi_{acc}^w$, we have $\rho \models \phi$ and for every run $\rho \in \Pi_{rej}^w$, we have $\rho \not\models \phi$. Clearly $\Pi^w = \Pi_{acc}^w \cup \Pi_{rej}^w$ and $\Pi_{acc}^w \cap \Pi_{rej}^w = \emptyset$. If $w \in \mathcal{L}(\mathcal{P})$, it means that $Pr_{\mathcal{P}}(\Pi_{acc}^w) = 1 = 1 - Pr_{\mathcal{P}}(\Pi_{rej}^w)$. Then, $Pr_{\mathcal{P}}(\Pi_{rej}^w) = 0$. If we negate the acceptance condition, we know that

in the automaton $\bar{\mathcal{P}}$ with the same structure and acceptance condition $\neg\phi$, an accepting run ρ satisfies that $\rho \models \neg\phi$. Hence, $Pr_{\bar{\mathcal{P}}}(w) = Pr_{\bar{\mathcal{P}}}(II_{rej}^w) = 0$. Similarly, if $w \notin \mathcal{L}(\mathcal{P})$, we have $Pr_{\mathcal{P}}(II_{rej}^w) = 1 - Pr_{\mathcal{P}}(II_{acc}^w) = 1$. This then entails that $Pr_{\bar{\mathcal{P}}}(w) = Pr_{\bar{\mathcal{P}}}(II_{rej}^w) = 1$.

It then follows that $\mathcal{L}(\bar{\mathcal{P}}) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{P})$. \square

Proposition 2. *Let \mathcal{M} be an LMC, and \mathcal{P} be a 0/1-PA recognising the DBA language L . Then $Pr_s(\mathcal{L}(\mathcal{P}^q)) = \chi(\langle qs \rangle)$ for all $\langle qs \rangle \in Q \times S$. In particular, $Pr_{\mathcal{M}}(L) = Pr_{\mathcal{M} \times \mathcal{P}}(\Diamond K)$ where K is the set of all states in the accepting BSCCs of $\mathcal{P} \times \mathcal{M}$.*

Proof. Let $\mathbf{x}(\langle qs \rangle) = Pr_s(\mathcal{L}(\mathcal{P}^q))$ for all $s \in S$ and $q \in Q$. It suffices to show that \mathbf{x} is a solution of Eq. (1).

We abuse the probability function δ as a transition function such that $q' \in \delta(q, a)$ if $\delta(q, a)(q') > 0$.

For a product state $\langle ps \rangle$ in an accepting (resp. rejecting) BSCC, we have $\mathbf{x}(\langle ps \rangle) = 1$ (resp. $\mathbf{x}(\langle ps \rangle) = 0$). We prove this via a DBA \mathcal{D} which accepts the language L , that is, $\mathcal{L}(\mathcal{D}) = L$. Consider the product of $\mathcal{P} \times \mathcal{M} \times \mathcal{D}$ in which a transition $\langle psq \rangle \xrightarrow{a} \langle p's'q' \rangle$ is an α -transition if and only if $q \xrightarrow{a} q'$ is an α -transition in \mathcal{D} .

We make the following observations: (1) the BSCCs in $\mathcal{P} \times \mathcal{M} \times \mathcal{D}$ project onto BSCCs in $\mathcal{P} \times \mathcal{M}$ when the \mathcal{D} component is omitted; (2) for each BSCC in $\mathcal{P} \times \mathcal{M}$, there exists at least one BSCC in $\mathcal{P} \times \mathcal{M} \times \mathcal{D}$ that projects down to it; (3) for all states $\langle psq \rangle$ in $\mathcal{P} \times \mathcal{M} \times \mathcal{D}$, we have $\mathcal{L}(\mathcal{P}^p) = \mathcal{L}(\mathcal{D}^q)$; (4) for a state $\langle psq \rangle$ in an accepting (resp. rejecting) BSCC of $\mathcal{P} \times \mathcal{M} \times \mathcal{D}$, we have $\mathbf{x}(\langle ps \rangle) = 1$ (resp. $\mathbf{x}(\langle ps \rangle) = 0$). It remains to show that an accepting (resp. rejecting) BSCC in $\mathcal{P} \times \mathcal{M} \times \mathcal{D}$ corresponds to an accepting (resp. rejecting) BSCC in $\mathcal{P} \times \mathcal{M}$.

It is easy to see that if $\langle psq \rangle \xrightarrow{a} \langle p's'q' \rangle$ is an α -transition in a BSCC of $\mathcal{P} \times \mathcal{M} \times \mathcal{D}$, then there must be an α -transition in the corresponding BSCC of $\mathcal{P} \times \mathcal{M}$, since we have $\mathcal{L}(\mathcal{P}^p) = \mathcal{L}(\mathcal{D}^q)$. If there are no α -transitions in a BSCC of $\mathcal{P} \times \mathcal{M} \times \mathcal{D}$, there should be no α -transitions in the corresponding BSCC of $\mathcal{P} \times \mathcal{M}$. Assume for contradiction that $\langle ps \rangle \xrightarrow{a} \langle p's' \rangle$ is an α -transition in $\mathcal{P} \times \mathcal{M}$. Since it is a rejecting BSCC in $\mathcal{P} \times \mathcal{M} \times \mathcal{D}$, the transition $\langle psq \rangle \xrightarrow{a} \langle p's'q' \rangle$ is not an α -transition, moreover, all paths (words) generated by the LMC from state s are rejected by \mathcal{D}^q . However, we have a word w generated from state s that is accepted by \mathcal{P}^p since $\langle ps \rangle \xrightarrow{a} \langle p's' \rangle$ is an α -transition, contradicting that $\mathcal{L}(\mathcal{P}^p) = \mathcal{L}(\mathcal{D}^q)$.

Next, we show that $x = B_\otimes x$. We have:

$$\begin{aligned}
\mathbf{x}(\langle qs \rangle) &= Pr_s(\mathcal{L}(\mathcal{P}^q)) \\
&= \sum_{a \in \Sigma} M(a)(s, s') \cdot Pr_{s'}\left(\bigcup_{q' \in \delta(q, a)} \mathcal{L}(\mathcal{P}^{q'})\right) \\
&= \sum_{a \in \Sigma} M(a)(s, s') \cdot Pr_{s'}(\mathcal{L}(\mathcal{P}^{q'})) \\
&\quad [\mathcal{P} \text{ is semantically deterministic and } q' \text{ is any } a\text{-successor of } q]
\end{aligned}$$

$$\begin{aligned}
&= \sum_{a \in \Sigma} M(a)(s, s') \cdot \sum_{q' \in \delta(q, a)} \delta(q, a)(q') Pr_{s'}(\mathcal{L}(\mathcal{P}^{q'})) \\
&\quad [\sum_{q' \in \delta(q, a)} \delta(q, a)(q') = 1 \text{ and } \mathcal{P} \text{ is semantically deterministic}] \\
&= \sum_{a \in \Sigma} \sum_{q' \in \delta(q, a)} M(a)(s, s') \cdot \delta(q, a)(q') \cdot Pr_{s'}(\mathcal{L}(\mathcal{P}^{q'})) \\
&= \sum_{a \in \Sigma} \sum_{q' \in \delta(q, a)} \delta_{\otimes}(\langle qs \rangle, a)(\langle q' s' \rangle) \cdot Pr_{s'}(\mathcal{L}(\mathcal{P}^{q'})) \\
&= \sum_{\langle q' s' \rangle \in Q \times S} B_{\otimes}(\langle qs \rangle, \langle q' s' \rangle) \cdot Pr_{s'}(\mathcal{L}(\mathcal{P}^{q'})) \\
&= \sum_{\langle q' s' \rangle \in Q \times S} B_{\otimes}(\langle qs \rangle, \langle q' s' \rangle) \cdot \mathbf{x}(\langle q' s' \rangle).
\end{aligned}$$

□

Lemma 3. *Let \mathcal{G} be an NCA that is semantically deterministic and safe deterministic and let R be a uniform stochastic resolver for \mathcal{G} . Then, the PA $\mathcal{G} \times R$ with Büchi acceptance condition is a 0/1-PA. Also, $\mathcal{L}(\mathcal{G} \times R) = \mathcal{L}(\mathcal{D})$.*

Proof. Let $\mathcal{G} = (\Sigma, Q, \delta, q_0, \alpha)$. Let R be a uniform stochastic resolver for \mathcal{G} , that is, $R(q, a)(q') = \frac{1}{|\delta_{\mathcal{G}}(q, a)|}$ for all $q \in Q_{\mathcal{G}}$, $a \in \Sigma$ and $q' \in \delta_{\mathcal{G}}(q, a)$. Now, we will prove that $\mathcal{G} \times R$ is a 0/1-PA for $\mathcal{L}(\mathcal{D})$ if we regard α -transitions as accepting under Büchi condition. Now we need to prove that $Pr_{\mathcal{G} \times R}(w) = 1$ for each $w \in \mathcal{L}(\mathcal{D})$ and $Pr_{\mathcal{G} \times R}(w) = 0$ for each $w \in \mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{C})$.

In the following, we do not distinguish the run of a word in $\mathcal{G} \times R$ and \mathcal{G} as they have one to one correspondence. If $w \in \mathcal{L}(\mathcal{D})$, i.e., $w \notin \mathcal{L}(\mathcal{G})$, there is no run in \mathcal{G} over w that gets trapped in a safe component. Hence, every run ρ of \mathcal{G} over w will visit α -transitions infinitely often, i.e., every run ρ is accepting in $\mathcal{G} \times R$. Clearly $Pr_{\mathcal{G} \times R}(w) = Pr_{\mathcal{G} \times R}^w(\{\pi : \pi \text{ is an accepting run over } w\}) = Pr_{\mathcal{G} \times R}^w(\{\pi : \pi \text{ is a run over } w\}) = 1$.

Now for a word $w \notin \mathcal{L}(\mathcal{D})$, i.e $w \in \mathcal{L}(\mathcal{G})$, assume by contradiction that the probability measure of accepting runs of w in $\mathcal{G} \times R$ is greater than 0. This means that there is a state q and $w = uw'$ for some $u \in \Sigma^*$, $w' \in \Sigma^\omega$ such that $q \in \delta(q_0, u)$ and the probability measure of accepting runs of w' from q in $\mathcal{G} \times R$ is 1. But then, since \mathcal{G} is semantically deterministic and $w \in \mathcal{L}(\mathcal{G})$, it can be shown by induction that for any prefix u' of w , any $q' \in \delta(q_0, u')$, w must have a continuation of run from q' to a safe component. Hence we must be able to find a finite run from q that leads to a deterministic safe component. This then entails that there is a positive probability, say ℓ , from q to a deterministic safe component, i.e., the probability measure of rejecting runs of w' in $\mathcal{G} \times R$ from q is greater than 0. This contradicts with the fact that the probability measure of accepting runs of w' from q is 1. Hence, the probability measure of accepting runs over w is not greater than 0. That is, we have $Pr_{\mathcal{G} \times R}(w) = 0$ for each word $w \notin \mathcal{L}(\mathcal{D})$. Hence this shows that $\mathcal{G} \times R$ is a 0/1 PA and $\mathcal{L}(\mathcal{D}) = \mathcal{L}(\mathcal{G} \times R)$. □

B Missing proofs in Section 4

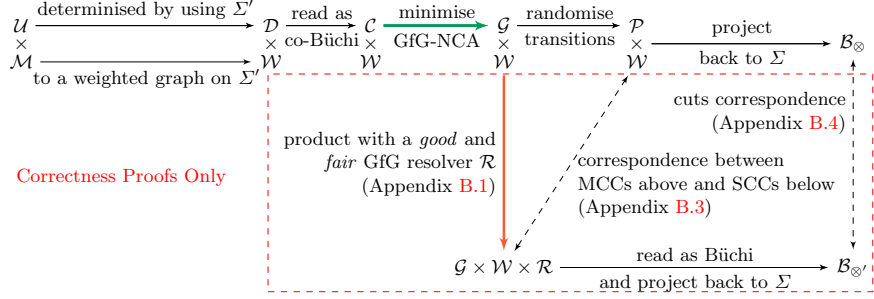


Fig. 3: The input \mathcal{U} is a UBA with state space Q and alphabet Σ . The input LMC \mathcal{M} also has the alphabet Σ . The steps in the algorithm or proof are represented by solid arrows, while correspondences between objects in the algorithm and proof are denoted with dashed arrows. A green arrow indicates a potential exponential state space reduction, whereas a red arrow signifies a significant blow-up (greater than exponential). However, this blow-up only occurs within the proof, not within the algorithm itself.

Below, we outline the main ideas, with the full proof presented later.

For qualitative model checking, \mathcal{U} accepts a random word generated by \mathcal{M} with positive probability if and only if a positive SCC exists in \mathcal{B}_{\otimes}' . As shown in Fig. 3, the upper row outlines our workflow, with the proof provided below it. We establish Proposition 12 and Proposition 13 by proving that the existence of a positive MCC in $\mathcal{P} \times \mathcal{W}$ corresponds to a positive SCC in \mathcal{B}_{\otimes}' , and vice versa.

For quantitative model checking, the exact probabilities are determined by solving a linear equation system similar to Eq. (2), using the underlying matrix of \mathcal{B}_{\otimes}' and replacing MCCs of $\mathcal{P} \times \mathcal{W}$ with SCCs of \mathcal{B}_{\otimes}' .

The algorithm relies on the observation that an SCC of \mathcal{B}_{\otimes}' is positive if and only if it contains a cut—a set of states whose associated probabilities sum to one. As shown in [5], such a cut for a positive SCC can be computed in polynomial time and even in NC. We prove Lemma 14 by demonstrating that for a positive MCC in $\mathcal{P} \times \mathcal{W}$, we can compute a set K corresponding to a cut K' for the associated positive SCC in \mathcal{B}_{\otimes}' .

Given the large size (more than exponential) of the good and fair resolver \mathcal{R} , we construct and analyse the product \mathcal{B}_{\otimes}' only within the proof to ensure correctness for the product $\mathcal{P} \times \mathcal{W}$.

Proposition 11. *Let \mathcal{M} be an LMC and \mathcal{U} be a UBA. The vector χ satisfies $\mathbf{x} = B_{\otimes} \mathbf{x}$.*

Proof. We have:

$$\chi(\langle qs \rangle) \tag{3}$$

$$\begin{aligned}
&= \sum_{a \in \Sigma'} \sum_{\langle q's' \rangle \in \delta_{\otimes}(\langle qs \rangle, a)} \frac{1}{|\delta_{\mathcal{G}}(q, a)|} \cdot W(a)(s, s') \chi(\langle q's' \rangle) \quad [\text{Lemma 8}] \\
&= \sum_{\langle q's' \rangle \in Q \times S} B_{\otimes}(\langle qs \rangle, \langle q's' \rangle) \cdot \chi(\langle q's' \rangle).
\end{aligned}$$

□

B.1 Existence of a good and fair resolver for $\mathcal{G} \times \mathcal{W}$

We denote by $f(w)$ the run ρ_f induced by the strategy function f on $w \in \Sigma^\omega$. Note that deterministic automata are trivially GfG with the strategy function $f(u) = \delta(q_0, u)$ for $u \in \Sigma^*$.

A run of the resolver $\mathcal{R} = (\mathbb{M}, m_0, g)$ over an ω -word $w = a_0 a_1 \dots$, denoted as $\mathcal{R}(w)$, is a sequence of pairs of memory states and input states $(m_0, q_0)(m_1, q_1) \dots$ such that for all $i \geq 0$, $q_{i+1} \in \delta(q_i, a_i)$ and $g(m_i, q_i, a_i) = (m_{i+1}, q_{i+1})$. We denote by $\mathcal{R}(w)|_Q$ the sequence obtained from $\mathcal{R}(w)$ by projecting away the memory states. We say \mathcal{R} a *good* resolver of \mathcal{A} , if for every accepting word $w \in \mathcal{L}(\mathcal{A})$, the run $\mathcal{R}(w)|_Q$ is accepted by \mathcal{A} . In order to construct \mathcal{R} , we will make use of an *anytime restart* resolver $\mathcal{R}_{\mathcal{G}} = (\mathbb{M}_{\mathcal{G}}, m_0, g_{\mathcal{G}})$ of \mathcal{G} . We say $\mathcal{R}_{\mathcal{G}}$ is an anytime restart resolver of \mathcal{G} if it is a good resolver of \mathcal{G} and given current memory state m and an automaton state $p \in Q_{\mathcal{G}}$, $\mathcal{R}_{\mathcal{G}}$ can restart from a state $q \in Q_{\mathcal{G}}$ with $q \sim_{\mathcal{G}} p$. That is, $\mathcal{R}_{\mathcal{G}}$ can obtain a run for a word $w \in \mathcal{L}(\mathcal{G}^p)$ that visits α -transitions *finitely often* from every state q and the memory state m if $p \sim_{\mathcal{G}} q$. The construction of such an anytime restart resolver for \mathcal{G} can be found in Lemma 22. The core behaviour of our anytime restart resolver is otherwise fairly standard: if the run that is safe of the longest moves to state q , it moves to a state q' whose safe language includes the safe language of q (“safe deterministic” is used to identify q while “immediately safe” guarantees the existence of q'). Our model checking algorithm will make use of the following results.

Theorem 18 ([1]). *Given a GfG-NCA \mathcal{A} , there is a polynomial-time algorithm to compute a minimal GfG-NCA \mathcal{A}' with $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$. Moreover, \mathcal{A}' is semantically deterministic, safe deterministic, and immediately safe.*

Definition 19. *Let $\mathcal{R} = (\mathbb{M}, m_0, g)$ be a resolver for $\mathcal{G} \times \mathcal{W}$.*

- *We say that a resolver \mathcal{R} is good for $\mathcal{G} \times \mathcal{W}$ if, for every word $w \in \mathcal{L}(\mathcal{G})$ that is generated by \mathcal{W} , the run of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ on w visits an $\alpha_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$ -transition only finitely often.*
- *We say \mathcal{R} is a fair resolver of $\mathcal{G} \times \mathcal{W}$ if, for every ω -word $w = a_0 a_1 \dots$, if a transition $(\langle qsm \rangle, a, \langle q's'm' \rangle) \in \Delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$ has been visited infinitely often in the run on w , then, for all the transitions $(\langle qs \rangle, a, \langle q''s' \rangle) \in \Delta_{\mathcal{G} \times \mathcal{W}}$, some transition $(\langle qsm'' \rangle, a, \langle q''s'm''' \rangle) \in \Delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$ must be visited infinitely often as well.*

We describe the main idea how to obtain a good and fair resolver of $\mathcal{G} \times \mathcal{W}$. We assume that we already have an anytime restart resolver $\mathcal{R}_{\mathcal{G}} = (\mathbb{M}_{\mathcal{G}}, m_{0,\mathcal{G}}, g_{\mathcal{G}})$ of \mathcal{G} . We can construct \mathcal{R} by trying to visit all successors for every state p of \mathcal{G} and transition (s, a, s') of \mathcal{W} in a round-robin manner, and then following a good choice made by $\mathcal{R}_{\mathcal{G}}$. The round-robin traversal guarantees the fairness among all successors, but does not necessarily give a good resolver. Therefore, we also need to make some reasonable choices by following $\mathcal{R}_{\mathcal{G}}$ after the round-robin traversal. The intuition is that, for semantically deterministic, safe deterministic and immediately safe GfG-NCA, it is fine to make bad choices and to restart for finitely many times as long as the right choice is eventually made. As \mathcal{G} is a safe deterministic *co-Büchi* automaton, an accepting run will eventually enter a deterministic safe component without any α -transitions.

Lemma 20. *There exists a good and fair resolver \mathcal{R} for the product $\mathcal{G} \times \mathcal{W}$.*

Proof. We now formally define the resolver $\mathcal{R} = (\mathbb{M}, m_0, g)$ where $\mathbb{M}, m_0 \in \mathbb{M}$ and $g : \mathbb{M} \times Q_{\mathcal{G} \times \mathcal{W}} \times \Delta_{\mathcal{W}} \mapsto \mathbb{M} \times Q_{\mathcal{G} \times \mathcal{W}}$ are the components described as before. A memory state $m = (f, (m_{\mathcal{G}}, q_{\mathcal{G}})) \in \mathbb{M}$ consists of a map $f : Q_{\mathcal{G}} \times \Delta_{\mathcal{W}} \mapsto Q_{\mathcal{G}} \cup \{\circ, +\}$ memorising the last successor we visit in \mathcal{G} , and the current configuration $(m_{\mathcal{G}}, q_{\mathcal{G}})$ of the anytime restart resolver $\mathcal{R}_{\mathcal{G}}$ over the current input word generated by \mathcal{W} . Here \circ indicates that we have finished the current round-robin traversal, while $+$ means that we should start the next round-robin traversal.

Let $Q_{\mathcal{G}} = \{q_1, \dots, q_{|Q_{\mathcal{G}}|}\}$ be the set of states in \mathcal{G} . For a round-robin traversal, we now assume the states will be ordered according to their index numbers. That is, state q_i is smaller than state q_j if $i < j$. Therefore, for a state $p \in Q_{\mathcal{G}}$ and $a \in \Sigma'$, we let $S_{<} = d_1 \dots d_{\ell}$ be the ordered sequence of states of $S = \delta_{\mathcal{G}}(p, a)$. We define $\text{NEXTSUCC}(p, a, d_h) = d_{h+1}$ if $1 \leq h < \ell$, $\text{NEXTSUCC}(p, a, d_{\ell}) = \circ$ and $\text{NEXTSUCC}(p, a, +) = d_1$.

Initially, we let $m_0 = (f_0, (m_{0,\mathcal{G}}, q_{0,\mathcal{G}}))$ where $f_0(q, (s, a, s')) = +$ for all $(s, a, s') \in \Delta_{\mathcal{W}}$ and $q \in Q_{\mathcal{G}}$. For a given memory state $m = (f, (m_{\mathcal{G}}, q_{\mathcal{G}}))$, current product state $\langle qs \rangle$ and transition $(s, a, s') \in \Delta_{\mathcal{W}}$, we define $(m', \langle q's' \rangle) = g(m, \langle qs \rangle, (s, a, s'))$ with $m' = (f', (m'_{\mathcal{G}}, q'_{\mathcal{G}}))$ as follows.

1. $(m'_{\mathcal{G}}, q'_{\mathcal{G}}) = g_{\mathcal{G}}(m_{\mathcal{G}}, q_{\mathcal{G}})$. That is, we keep track of the last $Q_{\mathcal{G}}$ -state the anytime restart resolver $\mathcal{R}_{\mathcal{G}}$ of \mathcal{G} , in order to make good choices for the successors in future.
2. If $\delta_{\mathcal{G}}^{\bar{\alpha}}(q, a)$ is defined or $|\delta_{\mathcal{G}}^{\alpha}(q, a)| = 1$, we have $q' = \delta_{\mathcal{G}}(q, a)$ because this is the only way to follow the transition of \mathcal{G} . Since \mathcal{G} is safe deterministic, we know that $|\delta_{\mathcal{G}}^{\bar{\alpha}}(q, a)| = 1$ for all $q \in Q$ and $a \in \Sigma'$. We also let $f' = f$.
3. Otherwise $|\delta_{\mathcal{G}}^{\alpha}(q, a)| \geq 2$. Let $S = \delta_{\mathcal{G}}(q, a)$ and $d = f(q, (s, a, s'))$.
 - If $d \neq \circ$, it means that we have not yet finished the current round-robin traversal of the successors in S . So, we let $q' = \text{NEXTSUCC}(q, a, d)$ and define $f' = f[(q, (s, a, s')) \mapsto q']$ by only updating the most recent successor of $(q, (s, a, s'))$ to q' in f .
 - If $d = \circ$, it means that we just finished the round-robin traversal and we need to make a good choice of successors now. Since $\mathcal{R}_{\mathcal{G}}$ is an anytime restart resolver, we are able to eventually choose a good successor $q'' \in$

$\delta^\alpha(q, a)$ such that \mathcal{R} is able to produce an accepting run of \mathcal{G} over any suffix $w \in \mathcal{L}(\mathcal{G}^{q'_G})$. That is, if \mathcal{R}_G produces an accepting run from q'_G , it can generate an accepting run from q'' as well. The intuition is that since \mathcal{G} is semantically deterministic and $q_G \sim_G q$, this entails that $q'_G \sim_G p'$ for all $p' \in \delta^\alpha(q, a)$. Because \mathcal{R}_G is an anytime restart resolver of \mathcal{G} , so it is able to select a good successor $q'' \in \delta^\alpha(q, a)$. We then let $q' = q''$ and define $f' = f[(q, (s, a, s')) \mapsto +]$, indicating we are ready to start another round-robin traversal for q and (s, a, s') .

It is easy to see that \mathcal{R} is a fair resolver for $\mathcal{G} \times \mathcal{W}$. This is because if a product state $\langle qs \rangle$ and transition $(s, a, s') \in \Delta_{\mathcal{W}}$ are visited infinitely often along the run induced by \mathcal{R} , i.e., some $(\langle qsm \rangle, a, \langle q's'm' \rangle)$ is visited infinitely often, by definition, all a -successors of q paired with s' will be visited in a round-robin manner for infinitely many times as well. That is, for all the transitions $(\langle qs \rangle, a, \langle q''s' \rangle) \in \Delta_{\mathcal{G} \times \mathcal{W}}$, some transition $(\langle qsm'' \rangle, a, \langle q''s'm''' \rangle) \in \Delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$ must be visited infinitely often as well.

Now we show that \mathcal{R} is a good resolver for $\mathcal{G} \times \mathcal{W}$. For every accepting word $w = a_0a_1 \cdots \in \mathcal{L}(\mathcal{G})$ generated by \mathcal{W} , \mathcal{R}_G can obviously produce an accepting run $\rho_G = q_{0,G}q_{1,G} \cdots$ of \mathcal{G} over w , since \mathcal{R}_G is a good resolver for \mathcal{G} . Let $\rho = q_0q_1 \cdots$ be the sequence of Q_G -states induced from $\mathcal{R}(w)$ by projecting away the memory states and the states from \mathcal{W} . By recursively applying Proposition 21, we have that $q_i \sim_G q_{i,G}$ for all $i \geq 0$. Since \mathcal{R}_G is an anytime restart resolver and $q_{i+1} \in \delta_G(q_i, a_i)$, ρ must be an accepting run of \mathcal{G} as well. This in turn entails that the run $\mathcal{R}(w)|_{Q_{\mathcal{G} \times \mathcal{W}}}$ will only visit $\bar{\alpha}_{\mathcal{G} \times \mathcal{W}}$ -transitions from some time point. It follows that $\mathcal{R}(w)|_{Q_{\mathcal{G} \times \mathcal{W}}}$ is accepting in $\mathcal{G} \times \mathcal{W}$ (as a co-Büchi automaton). Therefore, \mathcal{R} is a good and fair resolver of $\mathcal{G} \times \mathcal{W}$. \square

To complete the construction of \mathcal{R} , we now show how to construct an anytime restart resolver \mathcal{R}_G for \mathcal{G} . This construction only works when \mathcal{G} is semantically deterministic, safe deterministic and immediately safe. The intuition for constructing such a resolver is that over an input word w , we order the runs of \mathcal{G} over w by moving the runs that see $\bar{\alpha}$ -transitions earlier to the left. An accepting run of \mathcal{G} (as a co-Büchi automaton) will eventually reach a time point after which only $\bar{\alpha}$ -transitions are visited. Therefore, an accepting run will either be moved to the leftmost position or there must be other accepting runs which are further left to it. Since \mathcal{G} is safe deterministic, i.e., all $\bar{\alpha}$ -transitions are deterministic, then the number of runs which are further left to an accepting run will never increase and eventually stabilise. The idea to choose a successor by \mathcal{R}_G for a state q and letter $a \in \Sigma'$ is to choose the leftmost successor (i.e., run) among all in $\delta_G(q, a)$. This is because the leftmost run will have a better chance to be accepting than their right counterparts.

Formally, the resolver $\mathcal{R}_G = (\mathbb{M}_G, m_{0,G}, g_G)$ will use a set of memory states $\mathbb{M}_G \subseteq Q_G^{\leq |Q_G|}$ in which each memory state comprises of an *ordered* set of reachable states; the initial memory state is $m_{0,G} = q_{0,G}$ where $q_{0,G}$ is the initial state of \mathcal{G} . We define the transition function $g_G : \mathbb{M}_G \times Q_G \times \Sigma' \mapsto \mathbb{M}_G \times Q_G$ below. Let current memory state of \mathcal{R}_G be $m_G = d_1 \cdots d_\ell$, current state be p , the input

letter be $a \in \Sigma'$, let $(m'_G, q') = g_G(m_G, q, a)$ ⁸. First, for each $i \in \{1, \dots, \ell\}$, we define d''_i as

- ? if $\delta^{\bar{\alpha}}(d_i, a)$ is not defined,
- ? if $\delta^{\bar{\alpha}}(d_i, a)$ appears among $d'_1 \dots d'_j$ for some $j < i$, and
- $\delta^{\bar{\alpha}}(d_i, a)$ otherwise.

Note that \mathcal{G} is safe deterministic according to Theorem 18. In other words, if $\delta^{\bar{\alpha}}(d_i, a)$ is defined, there must be only one successor. Intuitively, if a run has continuation via $\bar{\alpha}$ -transitions, then the run (encoded as its last state at the current step) will remain in the left positions and no new runs can be moved to its further left positions, as described in the third bullet point. When two runs join in the same state as explained in the second bullet point, we only need to keep the left run and cut off the other run. All runs that have been cut off will be marked with ? symbol. Previously, we have considered the successors from $\bar{\alpha}$ -transitions; Now, we consider the successors from α -transitions.

Let $d'' = d''_1 \dots d''_k$ be the sequence of states obtained from $d'_1 \dots d'_\ell$ by removing all ? symbols. Let $S_{<} = r_1 \dots r_h$ be the ordered sequence of states in $(\bigcup_{0 \leq i \leq \ell} \delta^{\alpha}(d_i, a)) \setminus \{d''_1, \dots, d''_k\}$, which is the set of successors not appearing in the previous computation. Intuitively, all the remaining runs that continue via α -transitions will be ordered according to their last states at the current step.

Now we let $m'_G = d'' \cdot r_1 \dots r_h$ and let q' be the state that occurs in the *leftmost* position in m'_G among states in $\delta(q, a)$. Note that here q can be any state from m_G , which is why \mathcal{R}_G can restart anytime since all states in m_G are equivalent. In fact, similar to [22], we use the Latest Appearance Record [7] of the determinisation procedure of NCAs to organise the runs in the m_G -component. The memory state m_G not only keeps track of the set of reachable states but also prioritises the runs that are most likely to be accepting to the leftmost. Hence, if there is an accepting run over the input word w , the run that consists of the leftmost possible successors selected by \mathcal{R}_G from m_G -sequence will be accepting as well.

Proposition 21 ([1]). *Let \mathcal{A} be a semantically deterministic automaton, $p, q \in Q$, $a \in \Sigma$ and transitions $(p, a, p'), (q, a, q') \in \Delta_{\mathcal{A}}$. If $p \sim_{\mathcal{A}} q$, then $p' \sim_{\mathcal{A}} q'$.*

Lemma 22. *There exists an anytime restart resolver \mathcal{R}_G for \mathcal{G} if \mathcal{G} is semantically deterministic, safe deterministic and immediately safe.*

Proof. We now need to show why it is an anytime restart resolver. Let $w = a_0 a_1 \dots \in \mathcal{L}(\mathcal{G})$. Let $\rho = q_0 q_1 \dots$ is an accepting run of \mathcal{G} over w . By recursively applying Proposition 21, since \mathcal{G} is semantically deterministic, for every $i \geq 0$, we have $q'_i \sim_{\mathcal{G}} q_i$ for all states q'_i in m_i where $\mathcal{R}_G(w) = (m_0, q'_0)(m_1, q'_1) \dots$ is the run of \mathcal{R}_G over w . Of course, q_i also belongs to m_i for all $i \geq 0$ since $m_0 = q_0$ and m_i contains all reachable states in $\delta(q_0, a_0 \dots a_{i-1})$ if $i > 0$. Since \mathcal{G} is safe deterministic, an accepting run will eventually enter a deterministic

⁸ Note that for every current memory state $m_G = d_1 \dots d_\ell$ and state q , we always have $q = d_i$ for some $1 \leq i \leq \ell$.

safe component. It follows that there exists some integer $k > 0$, such that $w_j = a_j a_{j+1} \dots \in \mathcal{L}_{safe}(\mathcal{G}^{q_j})$ for all $j \geq k$. Further, since \mathcal{G} is immediately safe, i.e., for every q'_i in m_i with $i \geq k$, since $w_{i+1} = a_{i+1} a_{i+2} \dots \in \mathcal{L}_{safe}(\mathcal{G}^{q_{i+1}})$, there is a successor $q''_{i+1} \in \delta_{\mathcal{G}}(q'_i, a_i)$ such that $\mathcal{L}_{safe}(\mathcal{G}^{q_{i+1}}) \subseteq \mathcal{L}_{safe}(\mathcal{G}^{q''_{i+1}})$. Therefore, after the time point k , no matter what q'_i is for $i \geq k$ (regardless the past choices), the resolver $\mathcal{R}_{\mathcal{G}}$ just chooses the leftmost successor in $\delta(q'_i, a_i)$ all the time. Since there must be a successor $q''_{i+1} \in \delta(q'_i, a_i)$ such that $w_{i+1} = a_{i+1} a_{i+2} \dots \in \mathcal{L}_{safe}(\mathcal{G}^{q''_{i+1}})$, then there is an accepting run ρ''_{i+1} from q''_{i+1} that will never visit α -transitions and only has one successor when reading the next letter. This run $q'_i \cdot \rho''_{i+1}$ either will never move to the leftmost position among all runs from q'_i , which means that the chosen run by $\mathcal{R}_{\mathcal{G}}$ is already accepting; or, it eventually moves to the leftmost position among the runs from q'_i and will be always chosen by $\mathcal{R}_{\mathcal{G}}$ afterwards. Either case will allow $\mathcal{R}_{\mathcal{G}}$ eventually be able to choose an accepting run $q'_{i+1} q'_{i+2} \dots$ for $w_{i+1} \in \mathcal{L}_{safe}(\mathcal{G}^{q_{i+1}})$. Hence, $\mathcal{R}_{\mathcal{G}}$ is a good resolver. Since $\mathcal{R}_{\mathcal{G}}$ is allowed to make finitely many bad choices and eventually make the good choices of successors some point after k , we can replace any state q'_{j+1} with $q''_{j+1} \in m_{j+1}$ in the construction and $\mathcal{R}_{\mathcal{G}}$ is still able to restart from q''_{j+1} and obtain an accepting run for a word $a_{j+1} a_{j+2} \dots \in \mathcal{L}(\mathcal{G}^{q'_{j+1}})$ from q''_{j+1} since $q'_{j+1} \sim_{\mathcal{G}} q''_{j+1}$. Therefore, $\mathcal{R}_{\mathcal{G}}$ is an anytime restart resolver. \square

B.2 Product on the proof side

With the good and fair resolver from Appendix B.1 at hand, we are ready to construct the product $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ over Σ' and $\mathcal{B}_{\otimes'}$ over Σ .

The product $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ over Σ' is defined as $(\Sigma', Q_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}, \delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}, \langle q_0 \mathcal{G} s_0 m_0 \rangle, W_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}, \alpha_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}})$ where

- $Q_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}} = Q_{\mathcal{G}} \times S \times \mathbb{M}$ is the set of product states,
- $\delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$ is the transition function for the product automaton such that $\langle q' s' m' \rangle \in \delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}(\langle q s m \rangle, a)$ for some $(s, a, s') \in \Delta_{\mathcal{W}}$ where $(m', \langle q' s' \rangle) = g(m, \langle q s \rangle, (s, a, s'))$,
- $\langle q_0 \mathcal{G} s_0 m_0 \rangle \in Q_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$ is the initial state of the product,
- $W_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}} : \Sigma' \mapsto \mathbb{R}^{Q_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}} \times Q_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}}$ is the weight function such that $W_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}(a)(\langle q s m \rangle, \langle q' s' m' \rangle) = W(a)(s, s')$ for all transitions $(\langle q s m \rangle, a, \langle q' s' m' \rangle) \in \Delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$, and
- $\alpha_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}} = \{(\langle q s m \rangle, a, \langle q' s' m' \rangle) \in \Delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}} \mid (q, a, q') \in \alpha_{\mathcal{G}}\}$ is the set of α -transitions, where $\Delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}} \subseteq Q_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}} \times \Sigma' \times Q_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$ is the set of transitions in $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$.

We project the alphabet down to Σ , and define the product WG over Σ as $\mathcal{B}_{\otimes'} = (\Sigma, Q_{\otimes'}, \delta_{\otimes'}, \langle q_0 \mathcal{G} s_0 m_0 \rangle, W_{\otimes'}, \alpha_{\otimes'})$ where

- $Q_{\otimes'} \subseteq Q_{\mathcal{G}} \times S \times \mathbb{M}$ is the set of product states,
- $\delta_{\otimes'}$ is the transition function for the product automaton such that $\langle q' s' m' \rangle \in \delta_{\otimes'}(\langle q s m \rangle, a)$ if there exist a' with $a = \mathbf{p}^{-1}(a')$ and $\langle q s m \rangle, \langle q' s' m' \rangle \in Q_{\otimes'}$ with $(\langle q s m \rangle, a', \langle q' s' m' \rangle) \in \Delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$,

- $\langle q_0 s_0 m_0 \rangle \in Q_{\otimes'}$ is the initial state of the product,
- $W_{\otimes'} : \Sigma \mapsto \mathbb{R}^{Q_{\otimes'} \times Q_{\otimes'}}$ is the weight function such that $W_{\otimes'}(a)(\langle qsm \rangle, \langle q's'm' \rangle) = \sum_{a': \mathbf{p}^{-1}(a')=a} W(a')(s, s')$ for all $(\langle qsm \rangle, a, \langle q's'm' \rangle) \in \Delta_{\otimes'}$, and
- $\alpha_{\otimes'} = \{(\langle qsm \rangle, a, \langle q's'm' \rangle) \in \Delta_{\otimes'} \mid (\langle qs \rangle, a, \langle q's' \rangle) \in \alpha_{\otimes}\}$ is the set of α -transitions, where $\Delta_{\otimes'} \subseteq Q_{\otimes'} \times \Sigma \times Q_{\otimes'}$ is the set of transitions in $\mathcal{B}_{\otimes'}$.

We write $B_{\otimes'} = \sum_{a \in \Sigma} W_{\otimes'}(a)$.

Lemma 23. $\mathcal{B}_{\otimes'}$ is unambiguous and diamond-free.

Proof. We first prove unambiguity of $\mathcal{B}_{\otimes'}$. Let $w = a_0 a_1 \dots \in \mathcal{L}(\mathcal{B}_{\otimes'})$ be an ω -word. It suffices to show that there is a unique run in $\mathcal{B}_{\otimes'}$ on w .

Let $\pi = s_0 s_1 \dots$ of the LMC \mathcal{M} be the unique infinite path that produces the word w .

We assume for contradiction that there are two different runs in $\mathcal{B}_{\otimes'}$ on w : $\rho = \langle q_0 s_0 m_0 \rangle a_0 \langle q_1 s_1 m_1 \rangle a_1 \dots$ and $\rho' = \langle q_0 s_0 m_0 \rangle a_0 \langle q'_1 s_1 m'_1 \rangle \dots$ of $\mathcal{B}_{\otimes'}$. Since ρ and ρ' are different, we can let $k > 0$ be the smallest integer such that $q_k \neq q'_k$ or $m_k \neq m'_k$. This entails that in the product $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ over Σ' , we have $(\langle q_{k-1} s_{k-1} m_{k-1} \rangle, b_1, \langle q_k s_k m_k \rangle)$ and $(\langle q_{k-1} s_{k-1} m_{k-1} \rangle, b_2, \langle q'_k s_k m'_k \rangle)$ for different letters $b_1, b_2 \in \Sigma'$, i.e., $b_1 \neq b_2$. This is because by the construction of \mathcal{R} (cf. the proof of Lemma 20 in Appendix B.1), we know that \mathcal{R} is deterministic, s.t. $b_1 = b_2$ entails $q'_k = q_k$ and $m_k = m'_k$ for a given q_{k-1} and a transition (s_{k-1}, b_1, s_k) . This further gives that there are two words $w_1, w_2 \in \Sigma'^\omega$ over which their runs in $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ are the same as ρ and ρ' except the letters on the transitions, that is, we have $w = \mathbf{p}^{-1}(w_1) = \mathbf{p}^{-1}(w_2)$. We have that there are two accepting runs in $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ on w_1 and w_2 , respectively. That is, these two runs visit $\alpha_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$ -transitions infinitely often, and it must be the case that $w_1 \notin \mathcal{L}(\mathcal{G})$, and $w_2 \notin \mathcal{L}(\mathcal{G})$, since \mathcal{R} is a good resolver.

Recall that $\mathcal{L}(\mathcal{D}) = \overline{\mathcal{L}(\mathcal{G})}$ (over Σ'). It follows that $w_1 \in \mathcal{L}(\mathcal{D})$ and $w_2 \in \mathcal{L}(\mathcal{D})$. Since $w_1 \neq w_2$, we have two different accepting runs in the DBA \mathcal{D} over the two different words. Together with $w = \mathbf{p}^{-1}(w_1) = \mathbf{p}^{-1}(w_2)$, this then gives two different accepting runs of \mathcal{U} over w since \mathcal{U} and \mathcal{D} are the same except the letters on transitions, contradicting with the fact that \mathcal{U} is a UBA.

Next, we show diamond-freeness of $\mathcal{B}_{\otimes'}$. Let $w = a_0 a_1 \dots a_{n-1} \in \Sigma^*$ be a finite word. Let $\pi = s_0 s_1 \dots s_n$ of the LMC \mathcal{M} be the unique finite path that produces the word w . Similarly, we can prove $\mathcal{B}_{\otimes'}$ is diamond-free by assuming for contradiction that there is a diamond in $\mathcal{B}_{\otimes'}$ for w , that is, two different runs $\rho = \langle q_0 s_0 m_0 \rangle a_0 \langle q_1 s_1 m_1 \rangle a_1 \dots a_{n-1} \langle q_n s_n m_n \rangle$ and $\rho' = \langle q_0 s_0 m_0 \rangle a_0 \langle q'_1 s_1 m'_1 \rangle \dots a_{n-1} \langle q'_n s_n m'_n \rangle$ of $\mathcal{B}_{\otimes'}$. Since ρ and ρ' are different, we can let $k > 0$ be the smallest integer such that $q_k \neq q'_k$ or $m_k \neq m'_k$. This entails that in the product $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ over Σ' , we have $(\langle q_{k-1} s_{k-1} m_{k-1} \rangle, b_1, \langle q_k s_k m_k \rangle)$ and $(\langle q_{k-1} s_{k-1} m_{k-1} \rangle, b_2, \langle q'_k s_k m'_k \rangle)$ for different letters $b_1, b_2 \in \Sigma'$, i.e., $b_1 \neq b_2$. This is because by the construction of \mathcal{R} in the proof of Lemma 20, we know that \mathcal{R} is deterministic, i.e., $b_1 = b_2$ indicates $q'_k = q_k$ and $m_k = m'_k$ for a given q_{k-1} and a transition (s_{k-1}, b_1, s_k) . This further gives that there are two words $w_1, w_2 \in \Sigma'^*$ over which their runs in $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ be the same as ρ and ρ' except

the letters on the transitions, that is, we have $w = \mathbf{p}^{-1}(w_1) = \mathbf{p}^{-1}(w_2)$. We take a word $w' \in \Sigma'^\omega$ in the language of the WG $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ starting from $\langle q_n s_n m_n \rangle$. Such a word exists because all states in $\mathcal{B}_{\otimes'}$ have nonempty languages. We have that there are two accepting runs in $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ on $w_1 w'$ and $w_2 w'$, respectively. That is, these two runs visit $\alpha_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$ -transitions infinitely often, it must be the case that $w_1 w' \notin \mathcal{L}(\mathcal{G})$, and $w_2 w' \notin \mathcal{L}(\mathcal{G})$, since \mathcal{R} is a good resolver.

Recall that $\mathcal{L}(\mathcal{D}) = \mathcal{L}(\mathcal{G})$ (over Σ'). It follows that $w_1 w' \in \mathcal{L}(\mathcal{D})$ and $w_2 w' \in \mathcal{L}(\mathcal{D})$. Since $w_1 \neq w_2$, we have two different accepting runs in the DBA \mathcal{D} over the two different words. Together with $w = \mathbf{p}^{-1}(w_1) = \mathbf{p}^{-1}(w_2)$, this then gives two different accepting runs of \mathcal{U} over $w\mathbf{p}^{-1}(w')$ since \mathcal{U} and \mathcal{D} are the same except the letters on transitions, contradicting with that \mathcal{U} is a UBA. \square

Define the vector $\zeta \in [0, 1]^{Q_{\otimes'}}$ by $\zeta(\langle qsm \rangle) = Pr_s(\mathcal{L}(\mathcal{B}_{\otimes'}^{\langle qsm \rangle}))$ where we read the product $\mathcal{B}_{\otimes'}$ as a Büchi automaton. As a direct consequence of the definitions of the two vectors, ζ and χ , and the resolver \mathcal{R} being good, we have:

Lemma 24. $\zeta(\langle qsm \rangle) = \chi(\langle qs \rangle)$ for all $\langle qsm \rangle \in Q_{\otimes'}$. Furthermore, for all $q, q' \in Q$ with $q \sim_{\mathcal{G}} q'$, $s \in S$, and $m, m' \in \mathbb{M}$ we have $\zeta(\langle qsm \rangle) = \zeta(\langle q'sm' \rangle)$.

As a direct consequence of Lemma 24, we have $\zeta(q_0) = \zeta(\langle q_0 s_0 m_0 \rangle) = \chi(\langle q_0 s_0 \rangle) = \chi(q'_0) = Pr_{\mathcal{M}}(\mathcal{L}(\mathcal{U}))$, where q_0 and q'_0 are initial states of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ and $\mathcal{P} \times \mathcal{W}$ respectively. The following results on the SCCs of $\mathcal{B}_{\otimes'}$ (thus, of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$) will be used for some proofs in the following. The results can be found in [5]. Specifically, (1) is from the paragraph below [5, Proposition 7], (2) and (3) are from [5, Lemma 8], and (4) is from [5, Proposition 13]. The spectral radius of a matrix $M \in \mathbb{R}^{S \times S}$, denoted $\rho(M)$, is the largest absolute value of the eigenvalues of M .

Proposition 25. Let $(C', \Delta_{C'})$ be an SCC in $\mathcal{B}_{\otimes'}$, then

- (1) $\rho(B_{\otimes'}^{C', C'}) \leq 1$;
- (2) if $(C', \Delta_{C'})$ is recurrent, that is, $\rho(B_{\otimes'}^{C', C'}) = 1$, then $B_{\otimes'}^{C', C'} \zeta_{C'} = \zeta_{C'}$;
- (3) if $(C', \Delta_{C'})$ is recurrent, for all $c \in C'$, we have $\zeta(c) > 0$ iff $(C', \Delta_{C'})$ is accepting;
- (4) if $(C', \Delta_{C'})$ is recurrent and accepting, we have $\zeta(c) = 0$ for all $c \notin C'$ that is reachable from C' .

Furthermore, by [5, Lemma 12], for the product $\mathcal{B}_{\otimes'}$, we have:

Theorem 26 ([5]). The vector ζ uniquely solves the following system of linear equations:

$$\begin{aligned} \mathbf{z} &= B_{\otimes'} \mathbf{z} \\ \text{for all positive SCCs } (C', E') : \quad &\mu_{C'}^T \mathbf{z}_{C'} = 1 \\ &[\text{where } \mu_{C'} \text{ is the cut vector for } (C', E')] \\ \text{for all states } c' \text{ that cannot reach a positive SCC:} \quad &\mathbf{z}(c') = 0. \end{aligned} \tag{4}$$

Moreover, $\zeta(q_0) = Pr_{\mathcal{M}}(\mathcal{L}(\mathcal{U}))$ where q_0 is the initial state of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$.

B.3 Qualitative model checking

Proposition 25 shows that $Pr_{\mathcal{M}}(\mathcal{L}(\mathcal{U})) > 0$ iff the product $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ has a positive SCC. To prove that $Pr_{\mathcal{M}}(\mathcal{L}(\mathcal{U})) > 0$ iff the product $\mathcal{P} \times \mathcal{W}$ has a positive MCC, it suffices to demonstrate that the existence of a positive SCC in $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ implies the presence of a positive MCC in $\mathcal{P} \times \mathcal{W}$, and conversely.

Recall that $\Delta_{\mathcal{P} \times \mathcal{W}}$ (resp. $\Delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$) is the set of transitions in $\mathcal{P} \times \mathcal{W}$ (resp. $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$). We write $\text{free} : Q_{\mathcal{G}} \times S \times \mathbb{M} \rightarrow Q_{\mathcal{G}} \times S$ for the projection map from states in $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ (or $\mathcal{B}_{\otimes'}$) to states in $\mathcal{P} \times \mathcal{W}$ (or $\mathcal{G} \times \mathcal{W}$ and \mathcal{B}_{\otimes}): $\text{free}(\langle qsm \rangle) = \langle qs \rangle$. We abuse the notation and write $\text{free}(\langle \langle qsm \rangle, a, \langle q's'm' \rangle \rangle) = (\langle qs \rangle, a, \langle q's' \rangle)$. We extend free to $C \subseteq Q_{\mathcal{G}} \times S \times \mathbb{M}$ and to $\Delta \subseteq \Delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$ in the obvious manner.

We also introduce an arbitrary (but fixed) finest total preorder (tpo) \preceq on the states of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ that reflects their reachability relation. This means that we require the tpo to satisfy that, if $\langle qsm \rangle$ is reachable from $\langle q's'm' \rangle$, then $\langle q's'm' \rangle \preceq \langle qsm \rangle$ holds. Finest simply means that $\langle qsm \rangle \simeq \langle q's'm' \rangle$ holds if, and only if, $\langle qsm \rangle$ and $\langle q's'm' \rangle$ are in the same SCC. Note that for two states that cannot reach each other, one must be strictly greater than the other with respect to \preceq .

In the following, we first relate SCCs of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ and the closed components of $\mathcal{P} \times \mathcal{W}$.

Lemma 27. *If $(C', \Delta_{C'})$ is an SCC in the graph of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$, then $(C, \Delta_C) = (\text{free}(C'), \text{free}(\Delta_{C'}))$ in $\mathcal{P} \times \mathcal{W}$ is a closed component.*

Proof. Let $(C', \Delta_{C'})$ be an SCC in $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$. Let $C = \text{free}(C')$ and $\Delta_C = \text{free}(\Delta_{C'})$, the projection of C' and $\Delta_{C'}$ in $\mathcal{P} \times \mathcal{W}$. The component (C, Δ_C) induced by C and Δ_C is clearly strongly connected, and we have to show that it is closed in $\mathcal{P} \times \mathcal{W}$.

We assume for contradiction that the component (C, Δ_C) is not closed. Then there exist $\langle qs \rangle, \langle q_1s' \rangle, \langle q_2s' \rangle \in Q_{\mathcal{G}} \times S$ and $a \in \Sigma'$ such that $(\langle qs \rangle, a, \langle q_1s' \rangle) \in \Delta_C$, $(\langle qs \rangle, a, \langle q_1s' \rangle), (\langle qs \rangle, a, \langle q_2s' \rangle) \in \Delta_{\mathcal{P} \times \mathcal{W}}$, but $(\langle qs \rangle, a, \langle q_2s' \rangle) \notin \Delta_C$. That is, the transition $(\langle qs \rangle, a, \langle q_2s' \rangle)$ exits the component (C, Δ_C) .

Since $(\langle qs \rangle, a, \langle q_1s' \rangle) \in \Delta_C$, there must be some memory states $m, m_1 \in \mathbb{M}$ such that $(\langle qsm \rangle, a, \langle q_1s'm_1 \rangle)$ is in the SCC C' , and there must be a path in $(C', \Delta_{C'})$ from $\langle qsm \rangle$ back to itself that starts with this transition. Since \mathcal{R} is a fair resolver of $\mathcal{G} \times \mathcal{W}$, by Definition 19, this loop in $(C', \Delta_{C'})$ —and indeed every loop that contains the transition $(\langle qsm \rangle, a, \langle q_1s'm_1 \rangle)$ —must also contain a transition $(\langle qsm' \rangle, a, \langle q_2s'm_2 \rangle)$, which implies $(\langle qs \rangle, a, \langle q_2s' \rangle) \in \Delta_C$ by its definition. We then have the desired contradiction. \square

We now establish an inverse projection from MCCs in $\mathcal{P} \times \mathcal{W}$ to SCCs of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$.

Lemma 28. *Let (C, Δ_C) be an MCC in $\mathcal{P} \times \mathcal{W}$, $C_{\mathcal{R}} = \{\langle qsm \rangle \in Q_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}} \mid \langle qs \rangle \in C\}$ the set of states that project to C , $\langle p_0s_0m_0 \rangle$ a maximally ordered state of $C_{\mathcal{R}}$ with respect to \preceq . Then there is an SCC $(C', \Delta_{C'})$ in $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ with $\langle p_0s_0m_0 \rangle \in C'$ such that $(\text{free}(C'), \text{free}(\Delta_{C'})) = (C, \Delta_C)$.*

Proof. We start with building a pair of states and transitions $(C'', \Delta_{C''})$ in $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ according to the following rules:

1. $\langle p_0 s_0 m_0 \rangle \in C''$;
2. if $(\langle psm \rangle, a, \langle p's'm' \rangle) \in \Delta_{C''}$, then $\langle p's'm' \rangle \in C''$;
3. if $\langle psm \rangle \in C''$, $(\langle ps \rangle, a, \langle p's' \rangle) \in \Delta_C$ is an $\bar{\alpha}_{\mathcal{G} \times \mathcal{W}}$ -transition in $\mathcal{G} \times \mathcal{W}$, and $(\langle psm \rangle, a, \langle p's'm' \rangle)$ is an $\bar{\alpha}_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$ -transition in $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$, then $(\langle psm \rangle, a, \langle p's'm' \rangle)$ is an $\bar{\alpha}_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$ -transition in $\Delta_{C''}$, and
4. if $\langle psm \rangle \in C''$, $(\langle psm \rangle, a, \langle p's'm' \rangle)$ is an $\alpha_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$ -transition in $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$, and $(\langle ps \rangle, a, \langle p's' \rangle) \in \Delta_C$ is an $\alpha_{\mathcal{G} \times \mathcal{W}}$ -transition in $\mathcal{G} \times \mathcal{W}$, then $(\langle psm \rangle, a, \langle p's'm' \rangle)$ is an $\alpha_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$ -transition in $\Delta_{C''}$.

Let $(C'', \Delta_{C''})$ be the smallest such pair. We observe that (3) and (4) only add outgoing transitions that project on Δ_C , and (2) only adds states reachable over those transitions, so that $(\text{free}(C''), \text{free}(\Delta_{C''}))$ is contained in (C, Δ_C) and $(C'', \Delta_{C''})$ is closed under successor as defined in (2), (3) and (4).

This also provides an argument that all states in C'' are not greater than $\langle p_0 s_0 m_0 \rangle$ in $\text{tpo} \preceq$ (as $\langle p_0 s_0 m_0 \rangle$ was chosen to be \preceq -maximal). At the same time, as the construction only adds outgoing transitions in (3) and (4) and states reachable from previously constructed states, there is a simple inductive argument that all states are not smaller than $\langle p_0 s_0 m_0 \rangle$ with respect to the $\text{tpo} \preceq$.

With this in mind, we concurrently build two paths, one in (C, Δ_C) starting from $\langle p_0 s_0 \rangle$, and one in $(C'', \Delta_{C''})$, starting from $\langle p_0 s_0 m_0 \rangle$, namely:

$$\begin{aligned} \rho_C: & \langle p_0 s_0 \rangle, a_0, \langle p_1 s_1 \rangle, a_1, \langle p_2 s_2 \rangle, a_2, \langle p_3 s_3 \rangle, \dots, \langle p_n s_n \rangle \text{ in } (C, \Delta_C) \text{ and} \\ \rho_{C''}: & \langle p_0 s_0 m_0 \rangle, a_0, \langle p_1 s_1 m_1 \rangle, a_1, \langle p_2 s_2 m_2 \rangle, a_2, \langle p_3 s_3 m_3 \rangle, \dots, \langle p_n s_n m_n \rangle \text{ in } \mathcal{G} \times \mathcal{W} \times \mathcal{R}. \end{aligned}$$

The goal we pursue in ρ_C is to cover all states and transitions in (C, Δ_C) . We use $\rho_{C''}$ to make sure that a path corresponding to ρ_C in $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ exists when nondeterminism needs to be resolved. To construct the two paths, our strategy is as follows: When we are in state $\langle p_i s_i \rangle$ and have selected a_i and s_{i+1} in ρ_C , we use $\rho_{C''}$ to determine the next state p_{i+1} .

We first observe that the rules we used to build $(C'', \Delta_{C''})$ ensure that the finite path $\rho_{C''}$ is entirely within $(C'', \Delta_{C''})$.

This leaves the problem open how to make sure that (C, Δ_C) is covered. To ensure this, we can simply select, when we come to a state $\langle p_i s_i \rangle$ in the path ρ_C , a pair of a_i and s_{i+1} within (C, Δ_C) that has not been selected from this state for the longest amount of time. We continue until all states and transitions in (C, Δ_C) are covered. Our construction procedure will make sure that ρ_C covers (C, Δ_C) .

Following this strategy, we assume for contradiction that (C, Δ_C) is not covered by the path in ρ_C . Then the path in ρ_C never terminates, and it has an infinity set (D, Δ_D) (those states and transitions that do occur infinitely often). It follows that (D, Δ_D) is strictly smaller than (C, Δ_C) .

Then there is a $(\langle ps \rangle, a, \langle p's' \rangle) \in \Delta_C \setminus \Delta_D$ with $\langle ps \rangle \in D$. By the definition of an infinity set, $\langle ps \rangle$ occurs infinitely often after the last (if any) occurrence of

this transition, and therefore we then pick infinitely often the pair of a and s' . Since \mathcal{R} is a fair resolver, this leads to infinitely often selecting p' when looking at the path $\rho_{C''}$. This leads to a contradiction since p' will be selected by $\rho_{C''}$.

We have already shown that all states in C'' are equivalent with respect to \preceq , so that $(C'', \Delta_{C''})$ is contained in some SCC $(C', \Delta_{C'})$ of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$. By Lemma 27, an SCC $(C', \Delta_{C'})$ projects on a closed component (E, Δ_E) , so (E, Δ_E) must contain the MCC (C, Δ_C) , because the projection of $(C'', \Delta_{C''})$ already contains (C, Δ_C) . By construction, the only closed component that contains an MCC is the MCC itself, so that $(C', \Delta_{C'})$ projects onto (C, Δ_C) . \square

We also show that all \preceq -maximal SCCs of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ project down to MCCs of $\mathcal{P} \times \mathcal{W}$:

Proposition 29. *If $(C', \Delta_{C'})$ in $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ is the corresponding \preceq -maximal SCC of a closed component (C, Δ_C) in $\mathcal{P} \times \mathcal{W}$, we have that $(\text{free}(C'), \text{free}(\Delta_{C'}))$ is an MCC.*

Proof. It is easy to see that $(\text{free}(C'), \text{free}(\Delta_{C'}))$ is an closed component that encloses (C, Δ_C) . We assume for contradiction that $(\text{free}(C'), \text{free}(\Delta_{C'}))$ is not maximal. Let (C_m, Δ_{C_m}) be the MCC that encloses $(\text{free}(C'), \text{free}(\Delta_{C'}))$. We distinguish two cases:

- If we are missing some states in $\text{free}(C')$, that is, there exist states $\langle qs \rangle \in \text{free}(C')$ and $\langle q's' \rangle \notin \text{free}(C')$ such that there is a loop begins with $(\langle qs \rangle, a, \langle q's' \rangle)$ in (C_m, Δ_{C_m}) . There should also be a transition from $\langle qsm \rangle$ in C' to $\langle q's'm \rangle$ of another SCC. Then $\langle q's'm \rangle$ can reach a state $\langle q''s''m'' \rangle$ where $\langle q''s'' \rangle \in C$, which contradicts that $(C', \Delta_{C'})$ is the \preceq -maximal SCC of (C, Δ_C) .
- Otherwise, we have $\text{free}(C') = C_m$ and we are missing some transitions $(\langle qs \rangle, a, \langle q's' \rangle)$ for states $\langle qs \rangle, \langle q's' \rangle \in \text{free}(C')$. This means for states $\langle qsm \rangle$ in C' , an a -transition leads it to a state $\langle q's'm' \rangle$ out of the SCC, which contradicts that $(C', \Delta_{C'})$ is the \preceq -maximal SCC of (C, Δ_C) as $\langle q's'm' \rangle$ can reach a state $\langle q''s''m'' \rangle$ where $\langle q''s'' \rangle \in C$. \square

The following can be easily obtained from [5]:

Proposition 30. *For a state $\langle qsm \rangle$ of $\mathcal{B}_{\otimes'}$ that cannot reach a positive SCC, we have $\zeta(\langle qsm \rangle) = 0$.*

Proof. For all SCCs (C', E') in $\mathcal{B}_{\otimes'}$, we have $\rho(B_{\otimes', C'}^{C', C'}) \leq 1$ by (1) of Proposition 25. We distinguish two cases.

- If $\langle qsm \rangle$ is in a non-accepting recurrent SCC, then $\zeta(\langle qsm \rangle) = 0$ by (3) of Proposition 25.
- Otherwise, $\langle qsm \rangle$ is in a non-recurrent SCC. Assume by contradiction that $\zeta(\langle qsm \rangle) > 0$. Since $\rho(B_{\otimes', C'}^{C', C'}) < 1$ and $\zeta_{C'}$ is a nonnegative nonzero vector, we have $B_{\otimes', Q_{\otimes'}}^{C', Q_{\otimes'}} \zeta = \zeta_{C'} > B_{\otimes', C'}^{C', C'} \zeta_{C'}$. This means that (C', E') can reach $\langle q's'm' \rangle$ of another SCC (C'', E'') with $\zeta(\langle q's'm' \rangle) > 0$. This SCC (C'', E'')

cannot be positive by assumption. It is also not non-accepting recurrent since $\zeta(\langle q's'm' \rangle) > 0$. Thus, this SCC (C'', E'') again is non-recurrent. By similar arguments, the non-recurrent SCC (C', E') will reach an infinitely sequence of non-accepting, non-recurrent SCCs which contradicts with the fact that $\mathcal{B}_{\otimes'}$ is finite. \square

We define a new equivalence relation \equiv on Q_{\otimes} such that $c \equiv d$ if and only if c and d can be reached simultaneously from some state by the same word $w \in \Sigma'$ in $\mathcal{P} \times \mathcal{W}$. Let $W_{\mathcal{P} \times \mathcal{W}} : \Sigma' \rightarrow \mathbb{R}^{Q_{\otimes} \times Q_{\otimes}}$ be the weight function of $\mathcal{P} \times \mathcal{W}$ such that $W_{\mathcal{P} \times \mathcal{W}}(a)(\langle qs \rangle, \langle q't \rangle) = \delta_{\mathcal{P}}(q, a)(q') \cdot W(a)(s, t)$ for all $a \in \Sigma'$ and $(q, a, q') \in \Delta_{\mathcal{G}}$. We consider the transition matrix $B_{\mathcal{P} \times \mathcal{W}} = \sum_{a \in \Sigma'} W_{\mathcal{P} \times \mathcal{W}}(a)$. Let $W_{\mathcal{P} \times \mathcal{W}}^{(C, E)}$ be the transition function restricted to the closed component (C, E) and $B_{\mathcal{P} \times \mathcal{W}}^{(C, E)} = \sum_{a \in \Sigma'} W_{\mathcal{P} \times \mathcal{W}}^{(C, E)}(a)$ be the corresponding matrix restricted to the closed component (C, E) .

Lemma 31. *Let $(C', \Delta_{C'})$ be an SCC of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ and $(C, \Delta_C) = (\text{free}(C'), \text{free}(\Delta_{C'}))$ be a closed component in $\mathcal{P} \times \mathcal{W}$. Let \mathbf{x} be a nonnegative solution to $\mathbf{x} = B_{\mathcal{P} \times \mathcal{W}}^{(C, \Delta_C)} \mathbf{x}$ and $\mathbf{x}(c) = \mathbf{x}(d)$ for all $c, d \in C$ such that $c \equiv d$. Let \mathbf{z} be its projection on C' , that is, $\mathbf{z}(\langle qsm \rangle) = \mathbf{x}(\langle qs \rangle)$ for all $\langle qsm \rangle \in C'$. Then we have $\mathbf{z} \geq B_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}^{(C', \Delta_{C'})} \mathbf{z}$.*

Proof. We consider the successors of $\langle qsm \rangle$ in $(C', \Delta_{C'})$, and note that, for every successor $\langle q's'm' \rangle$ in $(C', \Delta_{C'})$ reachable over an $\bar{\alpha}$ -transition, $\langle q's' \rangle$ is reachable from $\langle qs \rangle$ over an $\bar{\alpha}$ -transition in (C, Δ_C) , and the probability entry is the same.

For every successor $\langle q's'm' \rangle$ in $(C', \Delta_{C'})$ reachable over an α -transition with probability $p > 0$, there are outgoing α -transitions to states $\langle r_i s' \rangle$ in (C, Δ_C) , such that, for all r_i , we have that $r_i \sim q'$, and for some i , we have $q' = r_i$. The closedness of (C, Δ_C) and the existence of this successor with $r_i = q'$ entail that all possible successors are included, which in turn implies that their probabilities sum up to p . (Note that, for all of these successors $\langle r_i s' \rangle$, the entry in \mathbf{x} is the same as the entry for $\langle q's'm' \rangle$ in \mathbf{z} by Lemma 24.)

Since each state $\langle qs \rangle$ of (C, Δ_C) can map to several different states $\langle qsm \rangle$ of $(C', \Delta_{C'})$, this provides the inequality \geq for an arbitrarily chosen entry, and thus overall. \square

Lemma 32. *Let (C, Δ_C) be an MCC of $\mathcal{P} \times \mathcal{W}$. Let $(C', \Delta_{C'})$ be the corresponding \preceq -maximal SCC of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ with $(\text{free}(C'), \text{free}(\Delta_{C'})) = (C, \Delta_C)$. Let \mathbf{x} be a nonnegative solution to $\mathbf{x} = B_{\mathcal{P} \times \mathcal{W}}^{(C, \Delta_C)} \mathbf{x}$ and $\mathbf{x}(c) = \mathbf{x}(d)$ for all $c, d \in C$ such that $c \equiv d$. Let \mathbf{z} be its projection on C' , that is, $\mathbf{z}(\langle qsm \rangle) = \mathbf{x}(\langle qs \rangle)$ for all $\langle qsm \rangle \in C'$. Then we have $\mathbf{z} \leq B_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}^{(C', \Delta_{C'})} \mathbf{z}$.*

Proof. We now consider the successors of $\langle qsm \rangle$ in $(C'', \Delta_{C''})$, and note that the construction of $(C'', \Delta_{C''})$ implies that $\langle qs \rangle \in C$. Moreover, for every successor $\langle q's' \rangle$ in (C, Δ_C) reachable over an $\bar{\alpha}$ -transition, the construction of $(C'', \Delta_{C''})$ guarantees that some state $\langle q's'm' \rangle$ is reachable from $\langle qsm \rangle$ over a similar $\bar{\alpha}$ -transition in $(C'', \Delta_{C''})$ —and thus in $(C', \Delta_{C'})$ —and that the probability entry is the same as for the transition from $\langle qs \rangle$ to $\langle q's' \rangle$ in (C, Δ_C) .

For every successor $\langle q's'm' \rangle$ in $(C'', \Delta_{C''})$ reachable over an α -transition with probability $p > 0$, the construction of $(C'', \Delta_{C''})$ guarantees that there is an outgoing α -transition to $\langle q's' \rangle$ in (C, Δ_C) . With the closedness of (C, Δ_C) we also get that the probability of all successors in (C, Δ_C) that are reachable over the similar α -transition in (C, Δ_C) add up to p . (Note that, for all of these successors, the entry in \mathbf{x} is the same as the entry for $\langle q's'm' \rangle$ in \mathbf{z} by Lemma 24.)

This provides the inequality \leq for an arbitrarily chosen entry, and thus overall. \square

An accepting SCC (C', E') of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ corresponds to an accepting SCC $(C', \mathbf{p}^{-1}(E'))$ of \mathcal{B}_{\otimes} , that is, an accepting SCC (C', E') will not be pruned from $\mathcal{P} \times \mathcal{W}$, since all states in C' have words that can traverse infinitely many α -transitions. Similarly, an accepting MCC (C, E) of $\mathcal{P} \times \mathcal{W}$ corresponds to an accepting MCC $(C, \mathbf{p}^{-1}(E))$ of \mathcal{B}_{\otimes} , that is, an accepting MCC (C, E) will not be pruned from $\mathcal{P} \times \mathcal{W}$. The following proposition follows from Lemma 31 and Lemma 32:

Proposition 33. *Let (C, Δ_C) be an accepting MCC of $\mathcal{P} \times \mathcal{W}$. Let $(C', \Delta_{C'})$ be the corresponding accepting \preceq -maximal SCC of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ with $(\text{free}(C'), \text{free}(\Delta_{C'})) = (C, \Delta_C)$. Let \mathbf{x} be a nonnegative solution to $\mathbf{x} = B_{\mathcal{P} \times \mathcal{W}}^{(C, \Delta_C)} \mathbf{x}$ and $\mathbf{x}(c) = \mathbf{x}(d)$ for all $c, d \in C$ such that $c \equiv d$. Let \mathbf{z} be its projection on C' , that is, $\mathbf{z}(\langle qsm \rangle) = \mathbf{x}(\langle qs \rangle)$ for all $\langle qsm \rangle \in C'$. Then we have $\mathbf{z} = B_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}^{(C', \Delta_{C'})} \mathbf{z}$.*

Proposition 34. *Let (C, Δ_C) be an accepting MCC of $\mathcal{P} \times \mathcal{W}$. For all nonnegative vectors \mathbf{x} satisfying $\mathbf{x}(c) = \mathbf{x}(d)$ for all $c, d \in C$ such that $c \equiv d$, we have $B_{\mathcal{P} \times \mathcal{W}}^{(C, \Delta_C)} \mathbf{x} \leq \mathbf{x}$.*

Proof. Let (C, Δ_C) be an MCC of $\mathcal{P} \times \mathcal{W}$. Let $(C', \Delta_{C'})$ be the corresponding \preceq -maximal SCC of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ with $(\text{free}(C'), \text{free}(\Delta_{C'})) = (C, \Delta_C)$. Let \mathbf{x} be a nonnegative vector satisfying $\mathbf{x}(c) = \mathbf{x}(d)$ for all $c, d \in C$ such that $c \equiv d$. Assume for contradiction that $B_{\mathcal{P} \times \mathcal{W}}^{(C, \Delta_C)} \mathbf{x} > \mathbf{x}$. The vector \mathbf{x} is then nonzero. Let \mathbf{z} be the projection of \mathbf{x} on C' , that is, $\mathbf{z}(\langle qsm \rangle) = \mathbf{x}(\langle qs \rangle)$ for all $\langle qsm \rangle \in C'$. We have $B_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}^{(C', \Delta_{C'})} \mathbf{z} > \mathbf{z}$. This could be shown, similar to Proposition 33 by arguing the entry for each state $\langle qsm \rangle \in C'$. Since \mathbf{z} is nonnegative and nonzero, this means $\rho(B_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}^{(C', \Delta_{C'})}) = \rho(B_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}^{C', C'}) > 1$, which contradicts $\rho(B_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}^{C', C'}) \leq 1$ as stated by (1) of Proposition 25. \square

Proposition 35. *There is a positive MCC (C, Δ_C) of $\mathcal{P} \times \mathcal{W}$ iff there is a positive SCC $(C', \Delta_{C'})$ of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ that projects to it, that is, $(\text{free}(C'), \text{free}(\Delta_{C'})) = (C, \Delta_C)$.*

Proof. Let (C, Δ_C) be a positive MCC of $\mathcal{P} \times \mathcal{W}$. Let $(C', \Delta_{C'})$ be the \preceq -maximal SCC from Lemma 28 such that $(\text{free}(C'), \text{free}(\Delta_{C'})) = (C, \Delta_C)$. Since (C, Δ_C) is recurrent, there is a nonnegative and nonzero solution \mathbf{x} to $B_{\mathcal{P} \times \mathcal{W}}^{(C, \Delta_C)} \mathbf{x} = \mathbf{x}$ and $\mathbf{x}(c) = \mathbf{x}(d)$ for all $c, d \in C$ such that $c \equiv d$.

Let \mathbf{z} be the projection of \mathbf{x} on C' , that is, $\mathbf{z}(\langle qsm \rangle) = \mathbf{x}(\langle qs \rangle)$ for all $\langle qsm \rangle \in C'$. By Proposition 33, \mathbf{z} is a nonnegative and nonzero solution to $\mathbf{z} = B_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}^{(C', \Delta_{C'})} \mathbf{z}$. By (1) of Proposition 25, we have $\rho(B_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}^{(C', \Delta_{C'})}) = 1$, that is, $(C', \Delta_{C'})$ is recurrent. Thus, $(C', \Delta_{C'})$ is a positive SCC.

It remains to show that the existence of a positive SCC implies the existence of a positive MCC. Assume that there is a positive SCC $(C', \Delta_{C'})$ of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$. Let $(C, \Delta_C) = (\text{free}(C'), \text{free}(\Delta_{C'}))$. We show that $(C', \Delta_{C'})$ must be a \preceq -maximal SCC of (C, Δ_C) . By (3) of Proposition 25, we have $\zeta(\langle qsm \rangle) > 0$ for all $\langle qsm \rangle \in C'$. By [5, Proposition 13], $\zeta(\langle qsm \rangle) = 0$ for all $\langle qsm \rangle$ in the other SCCs that are reachable from C' . It follows from Lemma 24 that $\langle qs \rangle \notin C$ for all states $\langle qsm \rangle$ in the other SCCs that are reachable from C' . Thus, $(C', \Delta_{C'})$ is the corresponding \preceq -maximal SCC of (C, Δ_C) .

By Proposition 29, (C, Δ_C) is an MCC. From (2) of Proposition 25, since $(C', \Delta_{C'})$ is positive, we have $B_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}^{(C', \Delta_{C'})} \zeta_{C'} = \zeta_{C'}$. From (3) of Proposition 25, we have $\zeta(\langle qsm \rangle) > 0$ for all $\langle qsm \rangle \in C'$. That is, $\zeta_{C'}$ is strictly positive. The vector χ_C is then strictly positive by Lemma 24. Since (C, Δ_C) is an MCC and $\chi(\langle qs \rangle) = \chi(\langle q's' \rangle)$ for all $\langle qs \rangle \equiv \langle q's' \rangle$, from Proposition 34, we have $B_{\mathcal{P} \times \mathcal{W}}^{(C, \Delta_C)} \chi_C \leq \chi_C$. Moreover, we have $\chi_C = B_{\mathcal{P} \times \mathcal{W}}^{C, Q_{\mathcal{P} \times \mathcal{W}}} \chi \geq B_{\mathcal{P} \times \mathcal{W}}^{(C, \Delta_C)} \chi_C$. Thus, $B_{\mathcal{P} \times \mathcal{W}}^{(C, \Delta_C)} \chi_C = \chi_C$. Along with (C, Δ_C) is accepting, we have (C, Δ_C) is a positive MCC. \square

Proposition 13 then follows from Proposition 25 and Proposition 35.

For a state $\langle qsm \rangle$ of $\mathcal{B}_{\otimes'}$ that cannot reach a positive SCC, we have $\zeta(\langle qsm \rangle) = 0$ by Proposition 30. Similarly, we have:

Proposition 12. *For a state $\langle qs \rangle$ of $\mathcal{P} \times \mathcal{W}$ that cannot reach a positive MCC, we have $\chi(\langle qs \rangle) = 0$.*

Proof. For a state $\langle qs \rangle$ of \mathcal{B}_{\otimes} that cannot reach a positive MCC, we consider two cases: either there exists a corresponding $\langle qsm \rangle$ in $\mathcal{B}_{\otimes'}$ or there does not. In the former case, we have $\chi(\langle qs \rangle) = 0$ by Lemma 24 and Proposition 30. In the latter case, we argue in the following that the values for such states are zero. As a simple consequence of Proposition 35, for a state $\langle qs \rangle$ in $\mathcal{P} \times \mathcal{W}$ (thus, \mathcal{B}_{\otimes}) that can reach a positive MCC, we have $\chi(\langle qs \rangle) > 0$. Moreover, $\mathcal{L}(\mathcal{B}_{\otimes'}^{(qsm)}) \neq \emptyset$; hence, $\langle qsm \rangle$ will not be pruned from $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$. Conversely, if $\langle qsm \rangle$ is pruned from $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$, that is, $\mathcal{L}(\mathcal{B}_{\otimes'}^{(qsm)}) = \emptyset$ and $\text{Pr}_s(\mathbf{p}^{-1}(\mathcal{L}(\mathcal{G}^q))) = 0$, then $\langle qs \rangle$ in $\mathcal{P} \times \mathcal{W}$ cannot reach a positive MCC. Thus, for a state $\langle qsm \rangle$ that is pruned from $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ and $\langle qs \rangle$ not pruned from $\mathcal{P} \times \mathcal{W}$, we have $\chi(\langle qs \rangle) = 0$. \square

B.4 Quantitative model checking

In the following, we describe the process for obtaining cut vectors for positive MCCs and demonstrate that, with the inclusion of the additional equations, the system (cf. Eq. (2)) has χ as its unique solution.

The following facts of the positive MCCs will help prove that Alg. 2 is a polynomial algorithm that computes a cut.

Lemma 36. *For a positive MCC (C, E) of $\mathcal{P} \times \mathcal{W}$, we have $B_{\otimes}^{(C, \mathbf{p}^{-1}(E))} \chi_C = \chi_C$.*

Proof. Recall that $\chi = B_{\otimes} \chi$. Thus, $\chi_C = B_{\otimes}^{C, Q_{\otimes}} \chi \geq B_{\otimes}^{(C, \mathbf{p}^{-1}(E))} \chi_C$. Since $\chi(c) = \chi(d)$ for all $c \equiv d$, we have $B_{\mathcal{P} \times \mathcal{W}}^{(C, E)} \chi_C \leq \chi_C$ by Proposition 34. Thus, $B_{\mathcal{P} \times \mathcal{W}}^{(C, E)} \chi_C = B_{\otimes}^{(C, \mathbf{p}^{-1}(E))} \chi_C = \chi_C$. \square

Lemma 37. *Let (C, E) be a positive MCC of $\mathcal{P} \times \mathcal{W}$. Then $\chi(\langle qs \rangle) > 0$ for all $\langle qs \rangle \in C$.*

Proof. By Proposition 35, there is a corresponding positive SCC (C', E') in $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ that projects down to the MCC (C, E) of $\mathcal{P} \times \mathcal{W}$. By [5, Lemma 8.1], we have $\zeta(\langle qsm \rangle) > 0$ for all $\langle qsm \rangle \in C'$. By Lemma 24, we have $\chi(\langle qs \rangle) = \zeta(\langle qsm \rangle)$ for all $\langle qsm \rangle \in Q_{\otimes}$. \square

Given a positive MCC (C, E) of $\mathcal{P} \times \mathcal{W}$, we have that, like SCCs, the positive MCC contains all transitions within C .

Lemma 38. *For a positive MCC (C, E) of $\mathcal{P} \times \mathcal{W}$, we have $B_{\otimes}^{C, C} = B_{\otimes}^{(C, \mathbf{p}^{-1}(E))}$.*

Proof. Let (C, E) be an MCC of $\mathcal{P} \times \mathcal{W}$. We have $B_{\otimes}^{C, C}(\langle qs \rangle, \langle q's' \rangle) \geq B_{\otimes}^{(C, \mathbf{p}^{-1}(E))}(\langle qs \rangle, \langle q's' \rangle)$ for all entries $(\langle qs \rangle, \langle q's' \rangle) \in Q_{\otimes} \times Q_{\otimes}$ and $B_{\otimes}^{(C, \mathbf{p}^{-1}(E))} = B_{\mathcal{P} \times \mathcal{W}}^{(C, E)}$.

Assume (C, E) is positive. By Lemma 36, we have $B_{\mathcal{P} \times \mathcal{W}}^{(C, E)} \chi_C = \chi_C$. By Lemma 37, χ_C is strictly positive.

We then have $B_{\otimes}^{C, C} \chi_C = B_{\mathcal{P} \times \mathcal{W}}^{C, C} \chi_C \geq B_{\mathcal{P} \times \mathcal{W}}^{(C, E)} \chi_C = \chi_C$. We also have $B_{\otimes}^{C, C} \chi_C \leq B_{\otimes}^{C, Q_{\otimes}} \chi = \chi_C$. This gives $B_{\otimes}^{C, C} \chi_C = \chi_C$. As χ_C is strictly positive, we have $B_{\otimes}^{C, C} = B_{\otimes}^{(C, \mathbf{p}^{-1}(E))}$. This completes the proof. \square

Lemma 39. *Let $\langle qs \rangle, \langle q's \rangle$ be two states in the positive MCC (C, E) with $\langle qs \rangle \sim_{\mathcal{P} \times \mathcal{W}} \langle q's \rangle$. For all $a' \in \Sigma'$ and all $\langle pt \rangle \in \delta_{\mathcal{P} \times \mathcal{W}}^C(\langle qs \rangle, a')$, there exists $\langle p't \rangle \in \delta_{\mathcal{P} \times \mathcal{W}}^C(\langle q's \rangle, a')$ with $\langle p't \rangle \sim_{\mathcal{P} \times \mathcal{W}} \langle pt \rangle$.*

Proof. To prove this lemma, we will use the corresponding \preceq -maximal positive SCC (C', E') . There are states $\langle qsm \rangle$ and $\langle q'sm' \rangle$ in C' . Since $\langle qs \rangle \sim_{\mathcal{P} \times \mathcal{W}} \langle q's \rangle$, we have $\langle qs \rangle \sim_{\mathcal{B}_{\otimes}} \langle q's \rangle$. Let $a' \in \Sigma'$ and assume there exists $\langle pt \rangle \in \delta_{\mathcal{P} \times \mathcal{W}}^C(\langle qs \rangle, a')$. There is a state $\langle ptm'' \rangle = \delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}^{C'}(\langle qsm \rangle, a')$, and $\zeta(\langle ptm'' \rangle) > 0$, by (3) of Proposition 25, since (C', E') is a positive SCC. As $\langle qs \rangle \sim_{\mathcal{B}_{\otimes}} \langle q's \rangle$, we have $\mathcal{L}((\mathcal{G} \times \mathcal{W} \times \mathcal{R})^{(qsm)}) = \mathcal{L}((\mathcal{G} \times \mathcal{W} \times \mathcal{R})^{(q'sm')})$. There must be a state $\langle p'tm''' \rangle = \delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}(\langle q'sm' \rangle, a')$ that has the same language as $\langle ptm'' \rangle$, meaning $\zeta(\langle p'tm''' \rangle) = \zeta(\langle ptm'' \rangle) > 0$. This state $\langle p'tm''' \rangle$ must be inside C' , because by (4) of Proposition 25, $\zeta(\langle p'tm''' \rangle) = 0$ if it is in a reachable SCC, which would be a contradiction. Thus, we have $\langle p't \rangle \in C$ and $\langle p't \rangle \in \delta_{\mathcal{P} \times \mathcal{W}}^C(\langle q's \rangle, a')$. \square

By applying Lemma 39 repeatedly, we have:

Lemma 40. Let $\langle qs \rangle, \langle q's \rangle$ be two states in the positive MCC (C, E) with $\langle qs \rangle \sim_{\mathcal{P} \times \mathcal{W}} \langle q's \rangle$. For all $w' \in \Sigma'^*$, $\langle pt \rangle \in \delta_{\mathcal{P} \times \mathcal{W}}^C(\langle qs \rangle, w')$ and $\langle p't \rangle \in \delta_{\mathcal{P} \times \mathcal{W}}^C(\langle q's \rangle, w')$, we have $\langle pt \rangle \sim_{\mathcal{P} \times \mathcal{W}} \langle p't \rangle$.

Lemma 41. Let $c_1 \sim_{\mathcal{P} \times \mathcal{W}} c_2$. Then $\delta_{\otimes}^C(c_1, w)/\sim_{\mathcal{P} \times \mathcal{W}} = \delta_{\otimes}^C(c_2, w)/\sim_{\mathcal{P} \times \mathcal{W}}$ for all $w \in \Sigma^*$.

Proof. Let $c_1 \sim_{\mathcal{P} \times \mathcal{W}} c_2$. Let $w \in \Sigma^*$ and $w' \in \Sigma'^*$ with $\mathbf{p}^{-1}(w') = w$. By Lemma 40, we have $d_1 \sim_{\mathcal{P} \times \mathcal{W}} d_2$ for all $d_1 \in \delta_{\mathcal{P} \times \mathcal{W}}^C(c_1, w')$ and all $\langle d_2 \rangle \in \delta_{\mathcal{P} \times \mathcal{W}}^C(c_2, w')$. Thus, $\delta_{\mathcal{P} \times \mathcal{W}}^C(c_1, w')/\sim_{\mathcal{P} \times \mathcal{W}} = \delta_{\mathcal{P} \times \mathcal{W}}^C(c_2, w')/\sim_{\mathcal{P} \times \mathcal{W}}$ and

$$\begin{aligned} \delta_{\otimes}^C(c_1, w)/\sim_{\mathcal{P} \times \mathcal{W}} &= \left(\bigcup_{w': \mathbf{p}^{-1}(w')=w} \delta_{\mathcal{P} \times \mathcal{W}}^C(c_1, w') \right) / \sim_{\mathcal{P} \times \mathcal{W}} \\ &= \left(\bigcup_{w': \mathbf{p}^{-1}(w')=w} \delta_{\mathcal{P} \times \mathcal{W}}^C(c_2, w') \right) / \sim_{\mathcal{P} \times \mathcal{W}} = \delta_{\otimes}^C(c_2, w)/\sim_{\mathcal{P} \times \mathcal{W}} \square \end{aligned}$$

Since $\mathcal{B}_{\otimes'}$ is unambiguous and diamond-free, all states of a cut of a positive SCC (C', E') of $\mathcal{B}_{\otimes'}$ are language disjoint, thus, their projections to \mathcal{B}_{\otimes} are all in different equivalence classes of $\sim_{\mathcal{B}_{\otimes}}$. Let $c' \in C'$ such that $\text{free}(c') = c$. For an arbitrary word $w \in \Sigma^*$, we will reach the same equivalence classes of $\sim_{\mathcal{B}_{\otimes}}$ when we start with $c' \in C'$ and $c \in C$.

Lemma 42. Let $c' \in C'$ such that $\text{free}(c') = c$. We have $\text{free}(\delta_{\otimes'}^{C'}(c', w)) \subseteq \delta_{\otimes}^C(c, w)$ and $\delta_{\otimes}^C(c, w)/\sim_{\mathcal{B}_{\otimes}} = \text{free}(\delta_{\otimes'}^{C'}(c', w))/\sim_{\mathcal{B}_{\otimes}}$ for all $w \in \Sigma^*$.

Proof. Let $c' \in C'$ such that $\text{free}(c') = c$. Let $w \in \Sigma^*$. It is easy to see that $\text{free}(\delta_{\otimes'}^{C'}(c', w)) \subseteq \delta_{\otimes}^C(c, w)$ since, by choice of the SCC C' , for all $w \in \Sigma^*$ and $\langle qsm \rangle \in \delta_{\otimes'}^{C'}(c', w)$ there is $\langle qs \rangle \in \delta_{\otimes}^C(c, w)$. We have $\text{free}(\delta_{\otimes'}^{C'}(c', w)) \subseteq \delta_{\otimes}^C(c, w)$.

We now consider the MCC (C, E) in the product of $\mathcal{P} \times \mathcal{W}$ (over Σ') and its corresponding \preceq -maximal SCC C' in the product $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ (over Σ'). Recall that $\delta_{\mathcal{P} \times \mathcal{W}}$ and $\delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}$ are the transition functions for $\mathcal{P} \times \mathcal{W}$ and $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ (over Σ'), respectively. Let $w' \in \Sigma'^*$ such that $\mathbf{p}^{-1}(w') = w$. Over the word w' , for any state reached from c in the MCC (C, E) of $\mathcal{P} \times \mathcal{W}$, there is a state that reached from c' in the SCC C' of $\mathcal{G} \times \mathcal{W} \times \mathcal{R}$ that recognises the same language over Σ' and vice versa. It follows that

$$\delta_{\mathcal{P} \times \mathcal{W}}^C(c, w')/\sim_{\mathcal{B}_{\otimes}} = \text{free}(\delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}^{C'}(c', w'))/\sim_{\mathcal{B}_{\otimes}}. \quad (5)$$

Thus, we have

$$\begin{aligned} \delta_{\otimes}^C(c, w)/\sim_{\mathcal{B}_{\otimes}} &= \left(\bigcup_{\mathbf{p}^{-1}(w')=w} \delta_{\mathcal{P} \times \mathcal{W}}^C(c, w') \right) / \sim_{\mathcal{B}_{\otimes}} \\ &= \left(\bigcup_{\mathbf{p}^{-1}(w')=w} \text{free}(\delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}^{C'}(c', w')) \right) / \sim_{\mathcal{B}_{\otimes}} \quad [\text{Eq. (5)}] \\ &= \text{free} \left(\bigcup_{\mathbf{p}^{-1}(w')=w} \delta_{\mathcal{G} \times \mathcal{W} \times \mathcal{R}}^{C'}(c', w') \right) / \sim_{\mathcal{B}_{\otimes}} \end{aligned}$$

$$\begin{aligned}
& [\text{swap free and the big union}] \\
& = \text{free}(\delta_{\otimes}^{C'}(c', w)) / \sim_{\mathcal{B}_{\otimes}}. \square
\end{aligned}$$

The following two lemmas, Lemma 43 and Lemma 44, ensure that Alg. 2 makes progress when the set $\delta_{\otimes}^C(c, w) / \sim_{\mathcal{P} \times \mathcal{W}}$ is not a cut and guarantee termination.

Lemma 43. *Suppose $w \in \Sigma^*$ is such that $c \in \delta_{\otimes}^C(c, w)$. If the set $\delta_{\otimes}^C(c, w) / \sim_{\mathcal{P} \times \mathcal{W}}$ is not a cut, then there are $v \in \Sigma^*$ and $d \not\sim_{\mathcal{P} \times \mathcal{W}} c$ with $\{c, d\} \subseteq \delta_{\otimes}^C(c, v)$ and $\delta_{\otimes}^C(d, w) \neq \emptyset$.*

Proof. By [5, Lemma 10], the positive SCC (C', E') has a cut. Therefore, there is $v \in \Sigma^*$ such that $K' = \delta_{\mathcal{B}_{\otimes}}^{C'}(c', v) \ni c'$ is a cut. Moreover, by Lemma 42, we have $\text{free}(K') \subseteq \delta_{\otimes}^C(c, v)$, and $\delta_{\otimes}^C(c, v) / \sim_{\mathcal{B}_{\otimes}}$ is a cut. Since $\text{free}(c') = c$, we have $c \in \delta_{\otimes}^C(c, v)$. Since $\delta_{\otimes}^C(c, vw) \supseteq \delta_{\otimes}^C(c, w)$, $\delta_{\otimes}^C(c, vw) / \sim_{\mathcal{B}_{\otimes}}$ is also a cut. Since $\delta_{\otimes}^C(c, vw) / \sim_{\mathcal{B}_{\otimes}}$ is a cut and $\delta_{\otimes}^C(c, w) / \sim_{\mathcal{B}_{\otimes}}$ is not, we have $\delta_{\otimes}^C(c, vw) / \sim_{\mathcal{B}_{\otimes}} \supsetneq \delta_{\otimes}^C(c, w) / \sim_{\mathcal{B}_{\otimes}}$.

Let $v_1 \in \Sigma'^*$ be the word satisfying $\mathbf{p}^{-1}(v_1) = v$ and $c \in \delta_{\mathcal{P} \times \mathcal{W}}^C(c, v_1)$. Assume for contradiction that starting from the state c there are no other words $v' \in \Sigma'^*$ in (C, E) such that $\mathbf{p}^{-1}(v') = v$. That is, $\delta_{\mathcal{P} \times \mathcal{W}}^C(c, v_1) = \delta_{\otimes}^C(c, v)$. For all states $d \in \delta_{\otimes}^C(c, v)$, we have $d \sim_{\mathcal{P} \times \mathcal{W}} c$ by definition of the relation $\sim_{\mathcal{P} \times \mathcal{W}}$, and $\delta_{\otimes}^C(c, w) / \sim_{\mathcal{P} \times \mathcal{W}} = \delta_{\otimes}^C(d, w) / \sim_{\mathcal{P} \times \mathcal{W}}$ by Lemma 41. Thus, $\delta_{\otimes}^C(c, w) / \sim_{\mathcal{B}_{\otimes}} = \delta_{\otimes}^C(d, w) / \sim_{\mathcal{B}_{\otimes}}$, and $\delta_{\otimes}^C(c, vw) / \sim_{\mathcal{B}_{\otimes}} = \delta_{\otimes}^C(c, w) / \sim_{\mathcal{B}_{\otimes}}$, which is a contradiction.

So there are $v_2 \in \Sigma'^*$ in (C, E) such that $v_2 \neq v_1$, $\mathbf{p}^{-1}(v_1) = \mathbf{p}^{-1}(v_2) = v$. By Lemma 8, for all $v_2 \neq v_1$, the complement language of c and d where d is any state in $\delta_{\mathcal{P} \times \mathcal{W}}^C(c, v_2)$ are disjoint, that is, $d \not\sim_{\mathcal{B}_{\otimes}} c$ for all $d \in \delta_{\mathcal{P} \times \mathcal{W}}^C(c, v_2)$. If for all such $v_2 \in \Sigma'^*$ in (C, E) we have $\delta_{\otimes}^C(d, w) = \emptyset$ for all $d \in \delta_{\mathcal{P} \times \mathcal{W}}^C(c, v_2)$, we have $\delta_{\otimes}^C(\delta_{\otimes}^C(c, v) \setminus c, w) = \emptyset$. Thus, $\delta_{\otimes}^C(c, vw) = \delta_{\otimes}^C(c, w) \cup \delta_{\otimes}^C(\delta_{\otimes}^C(c, v) \setminus c, w) = \delta_{\otimes}^C(c, w)$, a contradiction.

There must be a word $v_2 \in \Sigma'^*$ in (C, E) and a state $d \in \delta_{\mathcal{P} \times \mathcal{W}}^C(c, v_2)$ with $v_2 \neq v_1$, $\mathbf{p}^{-1}(v_1) = \mathbf{p}^{-1}(v_2) = v$ and $\delta_{\otimes}^C(d, w) \neq \emptyset$. Hence, with this word v , we have a state $d \not\sim_{\mathcal{P} \times \mathcal{W}} c$ with $\{c, d\} \subseteq \delta_{\otimes}^C(c, v)$ and $\delta_{\otimes}^C(d, w) \neq \emptyset$. \square

Lemma 44. *Let $w \in \Sigma^*$. Suppose there are a word $v \in \Sigma^*$ and a state $d \not\sim_{\mathcal{P} \times \mathcal{W}} c$ with $\{c, d\} \subseteq \delta_{\otimes}^C(c, v)$ and $\delta_{\otimes}^C(d, w) \neq \emptyset$. Then $\delta_{\otimes}^C(c, vw) / \sim_{\mathcal{B}_{\otimes}} \supsetneq \delta_{\otimes}^C(c, w) / \sim_{\mathcal{B}_{\otimes}}$.*

Proof. We use the unambiguity of the corresponding \preceq -maximal SCC (C', E') . Since $c \in \delta_{\otimes}^C(c, v)$, it holds that $\delta_{\otimes}^C(c, vw) \supseteq \delta_{\otimes}^C(c, w)$, and $\delta_{\otimes}^C(c, vw) / \sim_{\mathcal{B}_{\otimes}} \supseteq \delta_{\otimes}^C(c, w) / \sim_{\mathcal{B}_{\otimes}}$.

We show $c_1 \not\sim_{\mathcal{B}_{\otimes}} d_1$ for all $c_1 \in \delta_{\otimes}^C(c, w)$ and $d_1 \in \delta_{\otimes}^C(d, w)$. Such d_1 must exist as $\delta_{\otimes}^C(d, w) \neq \emptyset$. Let $c_1 \in \delta_{\otimes}^C(c, w)$ and $d_1 \in \delta_{\otimes}^C(d, w)$. Assume for contradiction that $c_1 \sim_{\mathcal{B}_{\otimes}} d_1$. By Lemma 42, there are $\{c'', d'\} \subseteq \delta_{\otimes}^{C'}(c', v)$, $c'_1 \in \delta_{\otimes}^{C'}(c', w)$ and $d'_1 \in \delta_{\otimes}^{C'}(d', w)$ such that $\text{free}(c'') \sim_{\mathcal{B}_{\otimes}} c$, $\text{free}(d') \sim_{\mathcal{B}_{\otimes}} d$, $\text{free}(c'_1) \sim_{\mathcal{B}_{\otimes}} c_1$ and $\text{free}(d'_1) \sim_{\mathcal{B}_{\otimes}} d_1$. If $c_1 \sim_{\mathcal{B}_{\otimes}} d_1$, then the languages of c'_1 and d'_1 will be the

same as well. Take a word $w' \in \Sigma^\omega$ in the language of c'_1 . Since $c \not\sim_{\mathcal{P} \times \mathcal{W}} d$, we have $c \not\sim_{\mathcal{B}_\otimes} d$, and $c' \neq d'$. Then, in $\mathcal{B}_{\otimes'}$, for the word $vw' \in \Sigma^\omega$, there are two accepting runs: one through v to c'' , then through w to c'_1 and the other run through v to d' , then through w to d'_1 . This contradicts the fact that $\mathcal{B}_{\otimes'}$ is unambiguous by Lemma 23. Thus, as $\delta_\otimes^C(c, vw) \supseteq \delta_\otimes^C(c, w) \cup \{d_1\}$, and $d_1 \not\sim_{\mathcal{B}_\otimes} c_1$ for all $c_1 \in \delta_\otimes^C(c, vw)$, we have $\delta_\otimes^C(c, vw)/\sim_{\mathcal{B}_\otimes} \not\supseteq \delta_\otimes^C(c, w)/\sim_{\mathcal{B}_\otimes}$. \square

Lemma 14. *Let C be a positive MCC. Then*

- (1) *C has a cut $K \subseteq C$ and $\mu_C^\top \chi_C = 1$, that is, $\sum_{k \in K} \chi(k) = 1$.*
- (2) *One can compute a cut K for C in polynomial time.*

Proof. If it does not hold that $\exists v \in \Sigma^*$ and $d \not\sim_{\mathcal{P} \times \mathcal{W}} c$ with $\{c, d\} \subseteq \delta_\otimes^C(c, v)$ and $\delta_\otimes^C(d, w) \neq \emptyset$, by Lemma 43 and Lemma 44, the word w gives the cut $\delta_\otimes^C(c, w)/\sim_{\mathcal{P} \times \mathcal{W}}$.

For the complexity of the algorithm, in every iteration, the set $\delta_\otimes^C(c, w)/\sim_{\mathcal{P} \times \mathcal{W}}$ increases by Lemma 44, so the algorithm terminates after at most $|C|$ iterations. By Lemma 43, the algorithm returns a cut.

Consider the directed graph $G = (V, E_V)$ with

$$\begin{aligned} V &= \{(p, p', s) \in Q_G \times Q_G \times S \mid \langle ps \rangle, \langle p's \rangle \in C\} \\ E_V &= \{(p, p', s) \rightarrow (q, q', t) \mid \exists a \in \Sigma \text{ such that } \langle qt \rangle \in \delta_\otimes^C(\langle ps \rangle, a) \\ &\quad \text{and } \langle q't \rangle \in \delta_\otimes^C(\langle p's \rangle, a)\} \end{aligned}$$

Let $c = \langle p_0 t_0 \rangle$. For any $v = v_1 \cdots v_n \in \Sigma^*$, a run of v in the LMC $t_0 v_1 t_1 \cdots v_n t_n$ and $c_n = \langle p_n s_n \rangle \in C$ we have $\delta_\otimes^C(c, v) \supseteq \{c_n, c'_n\}$ iff there are $p_1, p'_1, \dots, p_n, p'_n$ such that $(p_0, p_0, t_0) \rightarrow (p_1, p'_1, t_1) \rightarrow \cdots \rightarrow (p_n, p'_n, t_n)$ is a path in G . It follows that with a polynomial reachability analysis of G one can compute all $d \in C$ for which there exists $v_d \in \Sigma^*$ with $\{c, d\} \subseteq \delta_\otimes^C(c, v_d)$. The shortest such v_d corresponds to shortest paths in G , hence satisfy $|v_d| \leq |V| \leq |Q_G|^2 |S|$. Moreover, in polynomial time one can compute the equivalence relation $\sim_{\mathcal{P} \times \mathcal{W}}$, and check whether $d \sim_{\mathcal{P} \times \mathcal{W}} c$, one can also check in polynomial time whether $\delta_\otimes^C(d, w) \neq \emptyset$. This is because the equivalence relation $\sim_{\mathcal{P}}$ is given and it is polynomial time to compute $\sim_{\mathcal{W}}$ when we read \mathcal{W} as deterministic safe automaton. \square

The following propositions are key to prove that χ is the unique solution to Eq. (2):

Proposition 45. *For a positive MCC (C, E) of $\mathcal{P} \times \mathcal{W}$, we have $B_\otimes^{C, Q_\otimes} \chi = B_\otimes^{C, C} \chi_C = \chi_C$.*

Proof. By Lemma 36 and Lemma 38, we have $B_\otimes^{(C, \mathbf{p}^{-1}(E))} \chi_C = B_\otimes^{C, C} \chi_C = \chi_C$. We also have $\chi_C = B_\otimes^{C, Q_\otimes} \chi \geq B_\otimes^{C, C} \chi_C$. This completes the proof. \square

Proposition 46. *For a positive MCC (C, E) of $\mathcal{P} \times \mathcal{W}$ with a cut vector μ_C , χ_C is the unique solution to the linear equation system:*

$$\begin{aligned} B_{\otimes}^{C,C} \mathbf{x} &= \mathbf{x} \\ \mu_C^T \mathbf{x} &= 1 \end{aligned} \tag{6}$$

Proof. By Lemma 14 and Proposition 45, we have χ_C is a solution to the linear equation system Eq. (6). By (3) of Proposition 25 and Proposition 35, we have χ_C is strictly positive. Assume there is another nonnegative vector \mathbf{y} solves Eq. (6). Obviously, \mathbf{y} is nonzero.

We choose an entry $\langle qs \rangle \in C$ such that $f = \mathbf{y}(\langle qs \rangle) / \chi(\langle qs \rangle)$ is maximal. Since C is strongly connected, we have $\mathbf{y}(\langle q's' \rangle) = f \cdot \chi(\langle q's' \rangle)$ for all successors of $\langle q's' \rangle \in C$ of $\langle qs \rangle$. That is, \mathbf{y} is a scalar multiple of χ_C .

We have $\mu_C^T \chi_C = 1 = \mu_C^T \mathbf{y}$, hence $\chi_C = \mathbf{y}$. \square

We can now establish our main result on quantitative model checking.

Theorem 15. *(Quantitative MCMC) Given an LMC \mathcal{M} and a UBA \mathcal{U} , the linear equation system Eq. (2) has a unique solution \mathbf{v} . Moreover, we have $\mathbf{v} = \chi$, and thus $Pr_{\mathcal{M}}(\mathcal{L}(\mathcal{U})) = \mathbf{v}(q_0)$, where q_0 denotes the initial state in $\mathcal{P} \times \mathcal{W}$.*

Proof. First, we show that χ uniquely solves Eq. (2). The vector χ solves Eq. (2): this follows from Proposition 11, Proposition 12 and Lemma 14.

To show uniqueness, we assume for contradiction that there is a solution (nonnegative and nonzero) $\mathbf{y} \neq \chi$ to Eq. (2). For all states $\langle qs \rangle$ that cannot reach a positive MCC, we have $\chi(\langle qs \rangle) = \mathbf{y}(\langle qs \rangle) = 0$ by Proposition 12.

We select an entry $\langle qs \rangle$ in \mathbf{y} such that $\chi(\langle qs \rangle) > 0$, $\mathbf{y}(\langle qs \rangle) \neq \chi(\langle qs \rangle)$ and $f = \mathbf{y}(\langle qs \rangle) / \chi(\langle qs \rangle)$ is maximal. We have $f > 0$ and $f \neq 1$.

Such a state must exist and a positive MCC must be reachable from $\langle qs \rangle$ as $\chi(\langle qs \rangle) > 0$. We have that all states $\langle q's' \rangle$ that are reachable from $\langle qs \rangle$ in \mathcal{B}_{\otimes} also satisfy $\mathbf{y}(\langle q's' \rangle) = f \cdot \chi(\langle q's' \rangle)$. This can be shown by induction from the successors of $\langle qs \rangle$.

By Proposition 45 and Proposition 46, the values of the states $\langle q's' \rangle$ in a positive MCC are unique, that is, $\mathbf{y}(\langle q's' \rangle) = \chi(\langle q's' \rangle) > 0$, which contradicts that $\mathbf{y}(\langle q's' \rangle) = f \cdot \chi(\langle q's' \rangle) \neq \chi(\langle q's' \rangle)$.

Second, let q_0 be the initial state of $\mathcal{P} \times \mathcal{W}$, we have $\chi(q_0) = \chi(\langle q_0 g s_0 \rangle) = Pr_{s_0}(\mathbf{p}^{-1}(\mathcal{L}(\mathcal{P}))) = Pr_{s_0}(\mathcal{L}(\mathcal{U})) = Pr_{\mathcal{M}}(\mathcal{L}(\mathcal{U}))$, by Lemma 7. \square

C A family of UBAs with exponential gain

Here we will expand on the family of UBAs provided in Section 5 and prove Theorem 17. First, we recall the family of UBAs.

Definition A UBA $\mathcal{U}_n = (\Sigma, Q_n, \delta_n, \{q_0\}, \alpha_n)$ in the family consisting of

- the alphabet $\Sigma = \{\sigma, \pi, \#, \$\}$

- the set of states $Q_n = \{q_\$ \} \cup 2^{B_n}$ for $B_n = \{q_0, \dots, q_{n-1}\}$ and $\{q_0\}$ is the initial state
- the transition function δ_n producing the set of transitions $\Delta_n \subseteq Q_n \times \Sigma \times Q_n$
 - $\sigma : (P, \sigma, P') \in \Delta_n$ for $P' = \{q_{(i+1) \bmod n} \mid q_i \in P\}$,
 - $\pi : (P, \pi, P') \in \Delta_n$ for $P' = \{q_i \mid q_i \in P \wedge i \geq 2\} \cup \{q_{i \bmod 2} \mid q_i \in P \wedge i < 2\}$
 - $\# : (P, \#, P') \in \Delta_n$ for $P' = \{q_1, \dots, q_{n-1}\}$ if $P = \{q_0\}$ and $P \setminus \{q_0\}$ otherwise;
 - $\$: (P, a, q_\$) \in \Delta_n \forall P \in 2^{B_n}, a \in \Sigma \setminus \{\$, \}; (q_\$, \$, \{q_0\}) \in \Delta_n; (q_\$, \$, q_\$) \in \Delta$
- set of transitions $\alpha_n \subseteq \Delta_n$ describing the acceptance condition: $\alpha_n = \{(P_1, \#, P_2) \subseteq \Delta_n \mid q_0 \in P_1\} \cup \{(P, a, q_\$) \mid P \in 2^{B_n}\} \cup \{(q_\$, \$, \{q_0\}), (q_\$, \$, q_\$)\}$

The UBA \mathcal{U}_3 in this family is illustrated on the left of Fig. 1. Using the states $P \subseteq B_3$, \mathcal{U}_3 tracks a the set of tokens present in these boxes and follow them in the course of various actions until each of them is discarded. Then \mathcal{U}_n starts tracking a new set of tokens.

Let \mathcal{A}_n be the automata over $\Sigma_{\$} = \{\sigma, \pi, \#\}$ obtained by restriction of \mathcal{U}_n to the set of states $Q_n \setminus \{q_\$\}$ and Δ_n , α_n to transitions labelled by $\Sigma_{\$}$ over $Q_n \setminus \{q_\$\}$. Note that \mathcal{A}_n is deterministic.

For $q_i \in B_n$, let $\mathcal{L}_{safe}(\mathcal{A}_n^{\{q_i\}})$ be the set of infinite words which never encounters transitions from α_n when starting from $\{q_i\}$. Equivalently, these are the words under which the token from box q_i stays for an infinite duration. When \mathcal{A}_n is read as a deterministic co-Büchi automaton, the language of the DCA \mathcal{A}_n is $\mathcal{L}(\mathcal{A}_n) = \{ww' \in \Sigma^\omega \mid w' \in \mathcal{L}_{safe}(\mathcal{A}_n^{\{q_i\}}) \text{ for some } q_i\}$. When read as a DBA, its language is $\overline{\mathcal{L}(\mathcal{A}_n)} = \Sigma^\omega \setminus \mathcal{L}(\mathcal{A}_n)$.

$$\mathcal{L}(\mathcal{U}_n) = L_{\$}^\omega \cup L_n^*$$

where

$$L_{\$}^\omega = \{w \in \Sigma^\omega \mid w \text{ doesn't start with } \$ \text{ and has infinitely many } \$\text{'s}\} = \Sigma_{\$}^*(\Sigma^*\$)^\omega,$$

$$L_n^* = \{ww' \mid \text{where } w \in \Sigma^* \text{ and } w' \in \overline{\mathcal{L}(\mathcal{A}_n)}\}$$

The words in $L_{\$}^\omega$ contain infinitely many $\$$'s, whereas the words in L_n^* contain only finitely many $\$$'s and a suffix in $\overline{\mathcal{L}(\mathcal{A}_n)}$.

Lemma 47. *The automaton \mathcal{U}_n is unambiguous.*

Proof. Any valid run of a word in \mathcal{U}_n can encounter nondeterministic choices in two possible cases. Firstly, when from some state P and $a \in \Sigma \setminus \{\$, \}$, the choice is between $\delta_n(P, a)$ and $q_\$$. Since from $q_\$$ the only available transition is on $\$$, it follows that the transition to $q_\$$ is taken iff the letter following a is $\$$. In the second case, at $q_\$$ on reading letter $\$$, either it can stay on $q_\$$ or take transition to $\{q_0\}$. Again, since from $\{q_0\}$ the only valid transitions are on Σ , we can conclude that at $q_\$$ on letter $\$$ the transition to $\{q_\$\}$ is taken iff the letter following $\$$ is also $\$$. Since any word $w \in \mathcal{L}(\mathcal{U}_n)$ has a unique accepting run, \mathcal{U}_n is indeed a UBA. \square

Determinisation Alg. 3 determinises the UBAs \mathcal{U}_n by augmenting the alphabet Σ . We can use a map \mathbf{p} that makes it explicit in the actions from Σ if the next transition leads to $q_\$$. Since there are at most two nondeterministic choices at every state in \mathcal{U}_n , we can resolve the nondeterminism with a boolean variable. A value of 1 (denoting the next action is not $\$$) will direct a state to transition to the original DCA \mathcal{A}_n , while a value of 0 (denoting the next action is $\$$) will direct it to the state $q_\$$. The result is a DBA \mathcal{D}_n over $\Sigma' = \Sigma \times \{0, 1\}$. Since we will later treat \mathcal{D}_n as a DCA \mathcal{C}_n accepting the complement language, we make it complete by adding a new sink state, \perp . Specifically, we add $\bar{\alpha}$ (non-accepting for Büchi conditions) $\langle \$, 0 \rangle$ - and $\langle \$, 1 \rangle$ -transitions from P to \perp for all $P \in 2^{Q_n}$ and for the missing letters from $q_\$$ to \perp , and a $\bar{\alpha}$ (non-accepting for Büchi conditions) self-loop on \perp for all letters in Σ' . In this manner, we obtain the family of DCA $\mathcal{C}_n = (\Sigma', 2^{Q_n} \cup \{q_\$, \perp\}, \delta_{\mathcal{C}_n}, \{q_0\}, \alpha_{\mathcal{C}_n})$.

Essentially, in order to perform the action $\$$, first one needs to perform an action of the form $\langle a, 0 \rangle$ otherwise it is a wasted word that goes to the state \perp . Also, in the “all discard” state $q_\$$ one can only perform a discard action of the form $\langle \$, i \rangle$ and any other action will be wasted and go to the state \perp . $\mathcal{L}(\mathcal{C}_n)$ contains all words that are either wasted or under which some token stays infinitely. Let \mathcal{L}_\perp be the set of wasted word i.e. words that enter \perp after a finite duration. Hence $\mathcal{L}(\mathcal{C}_n) = \mathcal{L}_\perp \cup \left\{ w \in \Sigma'^\omega \mid w \text{ has a suffix in } \mathcal{L}_{safe}(\mathcal{C}_n^{q_i}) \text{ for some } q_i \right\}$.

Next we will define a family of NCAs that accept the same language as \mathcal{C}_n . These will turn out to be GfG-NCAs and exponentially smaller than \mathcal{C}_n 's.

Family of small GfG-NCA \mathcal{G}_n Here we define our family of NCAs $\{\mathcal{G}_n\}_{n \geq 2}$ with $\mathcal{G}_n = \langle \Sigma', Q_{\mathcal{G}_n}, q_0, \delta_{\mathcal{G}_n}, \alpha_{\mathcal{G}_n} \rangle$, where the set of states is $Q_{\mathcal{G}_n} = B_n \cup \{q_\$, \perp\}$. Intuitively instead of tracking a set tokens, we now track only one token at any instance. The transition function $\delta_{\mathcal{G}_n}$ is defined as follows:

- $\delta_{\mathcal{G}_n}(q_i, \langle \sigma, 1 \rangle) = \{q_{(i+1) \bmod n}\}$ for all $q_i \in B_n$,
- $\delta_{\mathcal{G}_n}(q_0, \langle \pi, 1 \rangle) = \{q_1\}$, $\delta_{\mathcal{G}_n}(q_1, \langle \pi, 1 \rangle) = \{q_0\}$, $\delta_{\mathcal{G}_n}(q_i, \langle \pi, 1 \rangle) = \{q_i\}$ for all $i \geq 2$,
- $\delta_{\mathcal{G}_n}(q_0, \langle \#, 1 \rangle) = B_n$, $\delta_{\mathcal{G}_n}(q_i, \langle \#, 1 \rangle) = \{q_i\}$ for all $i \geq 1$,
- $\delta_{\mathcal{G}_n}(q_\$, \langle \$, 1 \rangle) = \{q_0\}$,
- $\delta_{\mathcal{G}_n}(q_\$, \langle \$, 0 \rangle) = \{q_\$\}$,
- $\delta_{\mathcal{G}_n}(q_i, \langle a, 0 \rangle) = \{q_\$\}$ for all $q_i \in B_n$ and $a \in \Sigma$ and ,
- from every state, it transitions to \perp for the missing letters, that is, $\delta_{\mathcal{G}_n}(q, a) = \{\perp\}$ for all $q \in Q_{\mathcal{G}_n}$ and all the other letters $a \in \Sigma'$.

The set of transitions $\alpha_{\mathcal{G}_n}$ is defined to be $\{(q_0, \langle \#, 1 \rangle, q_i) \mid q_i \in B_n\} \cup \{(q_i, \langle a, 0 \rangle, q_\$) \mid a \in \Sigma\} \cup \{(q_\$, \langle \$, 0 \rangle, q_\$), (q_\$, \langle \$, 1 \rangle, q_0)\}$. Let $\mathcal{L}_\perp^{\mathcal{G}}$ denote the language of words that end up in state \perp in \mathcal{G}_n . Then the language accepted by NCA \mathcal{G}_n is $\mathcal{L}(\mathcal{G}_n) = \mathcal{L}_\perp^{\mathcal{G}} \cup \left\{ w \in \Sigma'^\omega \mid w \text{ has a suffix in } \mathcal{L}_{safe}(\mathcal{G}_n^{q_i}) \text{ for some } q_i \right\}$.

Lemma 48. *The following statements are true:*

- $\mathcal{L}(\mathcal{C}_n) = \mathcal{L}(\mathcal{G}_n)$.
- \mathcal{G}_n is a GfG automaton.

Proof. Recall that

$\mathcal{L}(\mathcal{C}_n) = \mathcal{L}_\perp \cup \left\{ w \in \Sigma'^\omega \mid w \text{ has a suffix in } \mathcal{L}_{\text{safe}}(\mathcal{C}_n^{\{q_i\}}) \text{ for some } q_i \right\}$. Since the safe component in the automaton restricted to states from $\{\{q_0\}, \dots, \{q_{n-1}\}\}$ in \mathcal{C}_n and the one restricted to states from B_n in \mathcal{G}_n are the same, we have $\mathcal{L}_{\text{safe}}(\mathcal{C}_n^{\{q_i\}}) = \mathcal{L}_{\text{safe}}(\mathcal{G}_n^{q_i})$. By the same argument the language we have $\mathcal{L}_\perp = \mathcal{L}_\perp^{\mathcal{G}}$. Finally, any run to states $\{q_i\}$ in \mathcal{C}_n can be simulated by a run to q_i in \mathcal{G}_n and vice versa. The same holds for \perp states in both automata. Hence we have $\mathcal{L}(\mathcal{C}_n) = \mathcal{L}(\mathcal{G}_n)$.

To show that \mathcal{G}_n is a GfG automaton observe that the only non-determinism is in state q_0 on reading $\langle \#, 1 \rangle$ which discards only the tracked token in box q_0 . One can design a resolver using the exact idea underlying the automaton \mathcal{C}_n where we track a set of non-discarded tokens. Even though each state of \mathcal{G}_n tracks a single token, in the resolver memory we track all non-discarded tokens from the point when \mathcal{G}_n started tracking the current token. Once this token gets discarded from q_0 , in \mathcal{G}_n we can start tracking another non-discarded token using the memory. Essentially this tracks all tokens that has stayed for the longest duration up to that point. Formally, the resolver is $\mathcal{R} = (\mathbb{M}, m_0, g)$ with $\mathbb{M} = Q_{\mathcal{C}_n}$, the initial memory state $m_0 = \{q_0\}$ and a partial memory function $g : \mathbb{M} \times B_n \times \Sigma'_\S \mapsto \mathbb{M} \times B_n$ defined below which is not valid on inputs of the form (P, q, a) where $P \subseteq B_n \wedge q \notin P$. Recall that $\delta_{\mathcal{C}_n}$ is the transition function of \mathcal{C}_n . Also since $\delta_{\mathcal{G}_n}$ is nondeterministic only at q_0 on input $\langle \#, 1 \rangle$, we abuse the notation for deterministic transitions by writing $\delta_{\mathcal{G}_n}(q, a)$ to signify the state G_n transitions to on input a from q .

$$\begin{aligned} & - g(P, q_0, \langle \#, 1 \rangle) = \\ & \quad \begin{cases} (P \setminus \{q_0\}, q_j) \text{ where } j = \min_i \{i \mid i > 0 \wedge q_i \in P\} & \text{if } P \subseteq B_n \wedge |P| > 1 \\ (B_n \setminus \{q_0\}, q_1) & \text{if } P = \{q_0\} \end{cases} \\ & - g(m, q, a) = (\delta_{\mathcal{C}_n}(m, a), \delta_{\mathcal{G}_n}(q, a)) \text{ for every other transition} \end{aligned}$$

Let $w = a_0 a_1 \dots \in \mathcal{L}(\mathcal{G}_n)$ and $\rho = q_0 \rho_1 \dots$ be an accepting run of \mathcal{G}_n over w . Let $(m_0, q_0)(m_1, \rho'_1) \dots$ be the run induced by \mathcal{R} and let $\rho' = q_0 \rho'_1 \dots$. By definition, there must be an integer $k > 0$ such that $(\rho_j, a_j, \rho_{j+1}) \notin \alpha_{\mathcal{G}_n}$ for all $j \geq k$. If ρ is accepting by reaching \perp , then ρ' will also reach \perp since the transition to \perp are similar and as a result become accepting. Also if ρ' is non-accepting by visiting q_\S infinitely often, then ρ also visits q_\S infinitely often, since the transitions to q_\S are deterministic. The only other way ρ can be accepting is if all transitions from ρ_k onwards are safe transitions in the part restricted to B_n . And the only way ρ' is non-accepting while staying in B_n is to read $\langle \#, 1 \rangle$ from q_0 infinitely often. Observe that from k onwards, for any two consecutive reading of $\langle \#, 1 \rangle$ from q_0 under ρ' , i.e. for $k_1 < k_2$, with $\rho'_{k_1} = \rho'_{k_2} = q_0$ and $a_{k_1} = a_{k_2} = \langle \#, 1 \rangle$ if $m_{k_1} \neq \{q_0\}$, we have $|m_{k_1}| > |m_{k_2}|$. Thus for some $h \geq k$, we would have $\rho'_h = q_0$, $a_h = \langle \#, 1 \rangle$ with $m_h = B_n \setminus \{q_0\}$. Since $a_h = \langle \#, 1 \rangle$, it follows that $\rho_h \neq q_0$. Hence, $\rho_h \in m_h$ and from definition of g , $\rho_j \in m_j \forall j \geq h$. Hence for some $h' > h$, we will have $\rho'_{h'} = q_0$, $a_{h'} = \langle \#, 1 \rangle$ and $\rho'_{h'+1} = \rho_{h'+1}$. But then from $h' + 1$ onwards the runs ρ and ρ' will be the same resulting in ρ' to be

an accepting run. Hence ρ' is an accepting run of w . Therefore, \mathcal{G}_n is a GfG automaton.

The construction details of the strategy is a simplified version of the construction of an anytime restart resolver in Appendix B.1. \square

Clearly \mathcal{G}_n is exponentially smaller than \mathcal{C}_n and \mathcal{U}_n . Thus, the minimal GfG-NCA recognising the same language as \mathcal{C}_n is exponentially smaller than \mathcal{C}_n , and thus \mathcal{U}_n . This proves Theorem 17.